An autonomous robot to clean heliostats

by

Tyron Jardine

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. W. Smit

December 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2020/09/01 .....

Abstract

An autonomous robot to clean heliostats

T. Jardine

Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (Mech)

December 2020

This thesis details the design and testing of a proof-of-concept robot capable of cleaning heliostats in concentrated solar power (CSP) plants. Heliostats accumulate dust, causing a decrease in their reflectivity, decreasing their ability to provide heat to the system. Current heliostat cleaning methods include large trucks which lack automation and use large amounts of water. The challenge was thus to design and build a robot capable of navigating heliostats of various sizes. The robot localises itself on the heliostat using a camera, ultrasonic sensors and wheel encoders. The camera makes use of Hough line detection to detect the edges of the heliostat and orientate the robot when near the middle of the mirror, and the ultrasonic sensors were used to detect when the robot was at the edges. The encoders ensure the robot can keep track of its movement.

A pinhole camera model was assumed for the camera because it has been calibrated. This allowed the 2D image co-ordinates of the heliostat edge lines to be transformed to real world 3D co-ordinates and to determine the orientation of the robot relative to the line. The extrinsic parameters of the pinhole camera model were derived both experimentally and theoretically, but the experimental values yielded more accurate line prediction results. It was seen that a higher camera resolution yielded better predictions of the robot's orientation, but the further away the robot was from the line the less capable it was of reliably determining its orientation. Higher Hough line thresholds also yielded less reliable results.

The robot could navigate the heliostat by making use of the ultrasonic sensors and wheel encoders when camera data was not available. The robot experienced problems with navigation at the heliostat corners and if the heliostat had short edges, but solutions to these problems are provided. A proof-

of-concept robot was thus designed, built, and tested that could navigate the surface of a heliostat using a camera, ultrasonic sensors, and wheel encoders. Artificial intelligence or machine learning can also be implemented with the camera to improve the system, but the robot has the potential to address the challenges faced in the existing cleaning methods of heliostats in large scale concentrated solar power plants.

Uittreksel

'n Outonome robot om heliostate skoon te maak

("An autonomous robot to clean heliostats")

T. Jardine

Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (Meg)

Desember 2020

Hierdie tesis beskryf die ontwerp en toets van 'n bewys van konsep robot wat heliostate in gekonsentreerde sonkragaanlegte kan skoonmaak. Heliostate versamel stof, wat die reflektiwiteit verminder en sodoende hul vermoë verminder om hitte aan die stelsel te verskaf. Huidige skoonmaakmetodes vir heliostate sluit groot vragmotors in wat min outomatisering het en wat groot hoeveelhede water gebruik. Die uitdaging was dus om 'n robot te ontwerp en te bou wat in staat is om heliostate van verskillende groottes te kan skoonmaak. Die robot lokaliseer homself op die heliostaat met behulp van 'n kamera, ultrasoniese sensors en wiel enkodeerders. Die kamera maak gebruik van Hough-lynopsporing om die rande van die heliostaat te vind en die robot te oriënteer wanneer dit naby aan die middel van die spieël is. Die ultrasoniese sensors word gebruik om te bepaal wanneer die robot naby aan die kante is. Die wiel enkodeerders help die robot se navigasie.

'n Speldgat-kameramodel word gebruik omdat die kamera gekalibreer is. Dit maak dit moontlik om die 2D-beeldkoördinate van die heliostaatrandlyne om te skakel na 3D-koördinate van die regte wêreld asook om die oriëntasie van die robot in verhouding tot die lyn te bepaal. Die ekstrinsieke veranderlikes van die speldgat-kameramodel is beide eksperimenteel en teoreties bepaal, maar die eksperimentele waardes het meer akkurate lynvoorspellingsresultate opgelewer. Daar is bevind dat 'n hoër kamera-resolusie beter voorspellings lewer van die oriëntasie van die robot, maar hoe verder die robot van die lyn af was, hoe swakker was die betroubaar daarvan. Hoër Hough-lyndrempels het ook minder betroubare resultate opgelewer.

Die robot kon oor die heliostaat navigeer deur gebruik te maak van die ultrasoniese sensors en wielkodeerders as kameradata nie beskikbaar was nie.

Die robot het problem ondervind om te navigeer wanneer dit naby aan die heliostathoeke was of wanneer die heliostatkante kort was, maar oplossings vir hierdie probleme word verskaf. 'n Bewys van die konsep-robot is dus ontwerp, gebou en getoets om die oppervlak van 'n heliostaat te navigeer met behulp van 'n kamera, ultrasoniese sensors en wielenkodeerders. Kunsmatige intelligensie of masjienleer kan ook met die kamera geïmplementeer word om die stelsel te verbeter. Die robot het die potensiaal om die uitdagings in die bestaande skoonmaakmetodes van heliostate in grootskaalse gekonsentreerde sonkragaanlegte aan te spreek.

Acknowledgements

I would like to thank Dr Willie Smit for his guidance and support in the writing of this thesis, as well as his unique approach at problem solving. Thank you for allowing me to take on this challenge.

I would also like to thank Dr Gerhard Venter, for convincing me to study further at the University of Stellenbosch, as well as for his friendship and guidance.

I would lastly like to thank my parents, for always making sure that this goal was achievable, I couldn't have done it without your continued support.

Dedications

This thesis is dedicated to all my family and friends that have supported me over the last two years. Motivation is not always a given, but I have received nothing but support from those around me. Thank you for always pushing me to do my best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Solar energy and the drive for sustainable development has created an increase in the construction of concentrated solar power plants (CSP) around the world. Concentrated solar power is a collection of technologies that focus the sun's energy at a collection point, allowing for process heat, heat storage or power generation. CSP is a relatively new technology that harvests solar energy but has been the focus for many researchers in the last few years.

The focus of research in CSP is broad, varying from its heat storage and chemicals to the calibration, orientation and cleaning of the heliostats [1]. One of the main advantages of CSP is that unlike other forms of renewable energy, the solar energy it absorbs can be stored as heat, allowing for energy supply during periods of low sunlight and ensuring higher energy efficiencies over systems such as photo-voltaic cells [2].

CSP plants, specifically the power tower configuration, use heliostats to redirect solar energy to the collection point, which are tall standing receiver surrounded by these heliostats [3]. Heliostats can track the movement of the sun to ensure that they are always redirecting sunlight at the central receiver. Extensive research is being done to ensure that heliostat tracking is as accurate as possible, as heliostats that are missing the receiver are not contributing to the heat generation of the system.

Heliostats exist in various shapes and sizes, but they all share the common problem of dust accumulation, resulting in a decrease of their reflectivity, or the amount of sunlight they can reflect. The drop in reflectivity, which can be as large as 25 percent after short outdoor exposure, when shared between many heliostats on a CSP plant causes high efficiency losses in the heat generation at the receiver [4]. The heliostats must be cleaned weekly to maintain high efficiencies but doing so in a field with a few thousand free-standing heliostats are a challenge most modern CSP plants experience. Figure 1.1 shows the effects soiling has on reflectance of a heliostat in an outdoor environment [4].

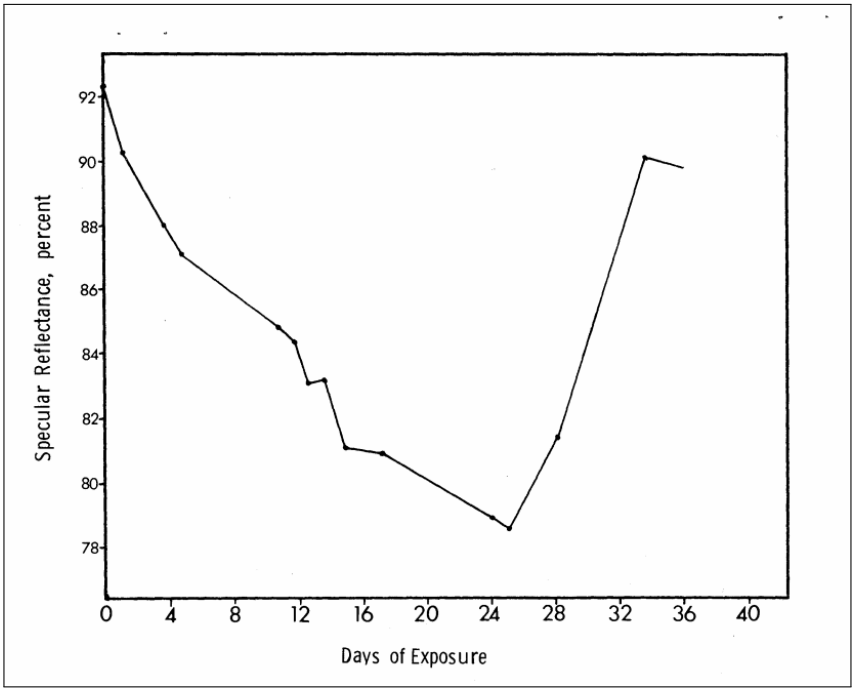


Figure 1.1: Heliostat reflectance vs time, when left in an outdoor environment for 8 weeks. The reflectance improvement are caused by rainfall over the testing period

The challenge was thus finding a cost effective and efficient means of cleaning these heliostats regularly. The current method of heliostat cleaning are the use of large trucks with high pressure hoses, filled with water and soap as in Figure 1.2 [5]. Although effective, these trucks are limited by human capability when navigating the spaces between heliostats as small errors often cause damages to the heliostats, and water is usually scarce in the dry locations where these plants are found. The heliostats are placed close together for maximum heliostat density, but roads must be made between them to allow these trucks to navigate. Additional to the trucks, a few robotic solutions have been developed in recent years to clean these heliostats [6], but their shortfalls prevent them from providing a permanent solution to the cleaning challenge. The largest shortfall being the inability of moving between heliostats, as they are often spaced with gaps between them. A fully autonomous cleaning robot needs to be capable of moving between heliostats.

Figure 1.2: Heliostat cleaning truck with high pressure hoses. These are the most used method of heliostat cleaning on most of the existing CSP plants

There is a need in the industry for a heliostat cleaning robot capable of cleaning the free-standing heliostats as seen in Figure 1.3 [7] and [8]. The largest challenge to these robots is localisation on the heliostat, and movement between the heliostats has not been solved with existing solutions. The thesis explores the design and testing of an autonomous proof-of-concept heliostat cleaning robot. The robot was designed to overcome the current shortfalls of the existing cleaning methods, which include high water usage and damages to heliostat from the cleaning trucks, and to provide a solution to move between free-standing heliostats which current robotic solutions lack. Additional to this, the challenge was to ensure the robot can localise itself on a heliostat while navigating safely and reliably. To overcome this challenge, the robot was fitted with a camera, and work on line detection was done to localise the robot on the heliostats. A magnetometer cannot be used, as the metal frames of the heliostats would cause interference.

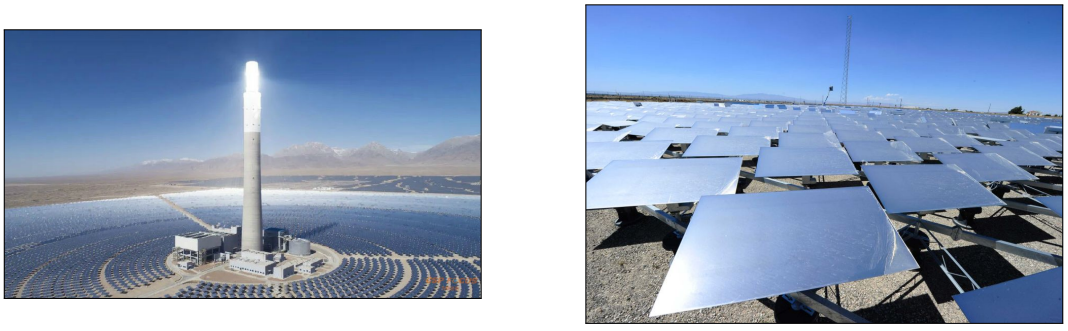


Figure 1.3: The Shouhang power tower in China, with a typical heliostat field. Most power tower CSP plants have free-standing heliostats that need to be cleaned regularly

The thesis details the design of a proof-of-concept heliostat cleaning robot, that aims to solve the challenge of localisation on a heliostat mirror. Red edges are used for testing of the concept to ensure the proof-of-concept robot was capable of differentiating the mirror edges from the surroundings. The robot makes use of a camera and Hough line detection to determine its orientation to the heliostat edges. In addition to the camera, ultrasonic sensors are used to detect when the robot has reached an edge, and wheel odometers are used to track relative motion. Figure 1.4 shows the robot during testing at the University of Stellenbosch.

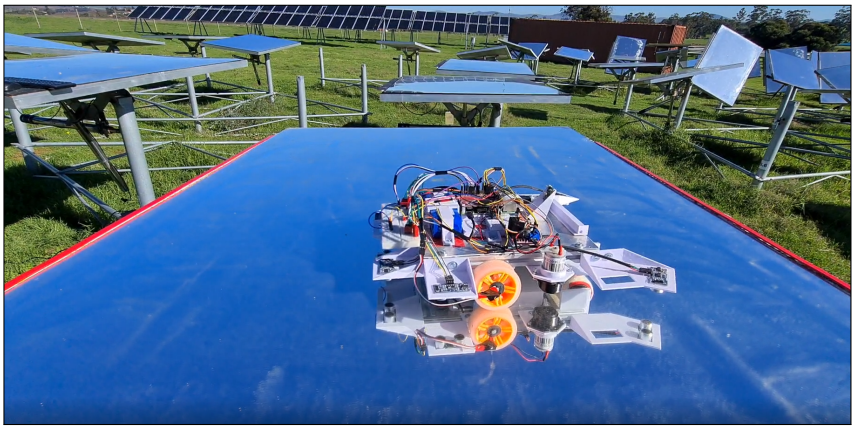


Figure 1.4: Proof-of-concept heliostat cleaning robot being tested at the University of Stellenbosch heliostat farm

The goals of this thesis, and the research are as follows:

- To design a proof-of-concept heliostat cleaning robot
- To build the heliostat cleaning robot
- To identify the edges of the heliostat with the camera
- To determine the robots orientation relative to the identified edges
- To localise and navigate the robot on the heliostat using the identified edges
- To determine the accuracy of the camera in predicting the robot's orientation when identifying edges, and the factors that affect the accuracy
- To discuss the possible shortfalls of the system and to provide solutions to them
- The research aims to determine whether robots may be a viable means of cleaning heliostats

- To determine the capability of Hough lines in heliostat edge detection applications
- To determine the components of future research to develop such a system for commercial implementation

The research methodology includes the formation of a literature study, which gathers information regarding current solutions to the heliostat cleaning problem as well as important components of robotic programming and hardware. The research then introduces the design process, starting with a set of requirements and specifications which are used to generate concepts for the robot. The concepts are then evaluated to obtain a detailed system design. The final design makes use of a camera to detect the heliostat edges in front of the robot, while ultrasonic sensors are used to prevent the robot from driving off an edge. Components of the systems navigation algorithm were discussed in detail in the design. Camera work was then done to analyse the accuracy of the camera when detecting the heliostat edges, by making use of both experimental and theoretical camera models, in combination with Hough line detection. The robot, its navigation algorithms and the camera were then tested on a heliostat to determine the capability of the system to navigate its surface.

Chapter 2

Literature Study

The chapter gives an overview of concepts that are central to the project. The chapter includes, amongst other things: an overview of CSP plants and heliostats, and the damage that cleaning methods may cause to the heliostat surfaces; and an explanation of the Hough line transform used in detecting lines in an image. The robotic operating system and robotic sensors are also discussed.

2.1 Concentrated Solar Power

Concentrated Solar Power, or CSP, is the concentration of solar energy towards a receiver. All CSP configurations make use of mirrors to focus this energy towards the receiver. The receiver collects the energy, where it is either stored, used for electricity production, or used as process heat in industrial applications [1]. Heat storage allows a CSP plant to meet energy demands when solar supply is low.

CSP has multiple configurations and are often paired with other forms of energy generation in hybrid plants [1]. The configurations include but are not limited to parabolic troughs, linear fresnels, power towers and parabolic dishes. Parabolic troughs are the earliest adaptation of commercialised CSP, while power towers are currently the fastest growing technology [9]. It is currently estimated that CSP generates around 1900 MW in the United States alone, with an estimated 10 000 MW being generated worldwide [10]. Along with the US, Spain, Chile and China are among the largest contributors to this production as shown in Figure 2.1 [10].

Although CSP is significantly cleaner than fossil fuel, the plants have high water demands that are difficult to meet in the dry regions where they operate. The high water demand is created by the need for cooling water, and the need to regularly clean the heliostats [11]. It is estimated that the Nevada Solar One CSP plant uses 400 acre-feet of water (490 megalitres) per annum whilst providing 50-64 MW of power [12]. In the same study it was found that the

Hualapai Valley Solar Project in Arizona was estimated to require 3000 acre-feet (3.7 gigalitres) of water per annum whilst providing 340 MW of power. The water demands induced within CSP plants are tremendous, and technologies which reduce water usage can provide significant improvements to CSP plants and their sustainability.

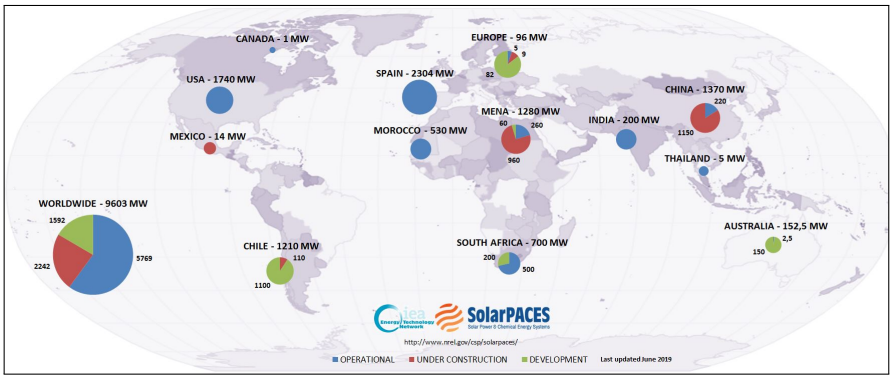


Figure 2.1: Locations of the largest CSP plants worldwide

2.2 Power Towers

Power tower CSP consists of a single, large receiver at the top of a tower. The towers are surrounded by heliostats which focus solar energy at the receiver. Most power towers make use of molten salts, which absorb heat at the receiver and act as the working fluid. Water has also been used as a working fluid in power tower CSP. The working fluids are used as thermal storage or passed through turbines in order to generate electricity [3].

Power towers are advantageous over parabolic troughs as the temperature of the working fluids are able to reach 1000° F (550° C), whereas that of parabolic troughs can reach only 750° F (400° C) [9]. A higher working temperature means higher energy production and a larger magnitude of thermal storage for the system, thermal efficiencies of up to 98 percent have been measured in these systems [13]. Power towers can be built on any piece of land, as flat ground is not required, although plant maintenance is easier on flat land [13]. Power towers are also scalable, the size of the plant can be adapted to suit the design load, and the molten salts are able to be reused without any thermal losses [14].

2.3 Heliostats

Heliostats are mirrors fixed to a rotating base that track the sun. The position of the Earth relative to the sun changes throughout the day, and as a result

the direction of sunlight changes with it. Heliostats counteract this change by adjusting their orientation to ensure the light is always focused at the receiver. This allows all the energy reflected from the sun to be received by the CSP plant receiver [15].

The sizes of heliostats vary from one square meter to a few hundred square meters, with the biggest advantage of size being the amount of energy reflected from the mirror. Although larger heliostats reflect more energy, the cost and ease of installation of smaller heliostats make them favourable for smaller CSP plants. Large heliostats are costly, difficult to install and maintain, but are favourable in large scale CSP plants due to their energy output [15]. A small scale CSP heliostat configuration can be seen below in Figure 2.2 [16].

Figure 2.2: Heliostats at the University of Stellenbosch Helio 100 site

The efficiency of heliostats has a direct impact on the overall system efficiency. The efficiency is determined by how accurately they redirect light, and the reflectivity of the mirror among other factors [17]. A study done by Sandina Labs in the USA found that heliostat efficiency losses can reach 25 percent after short periods of outdoor exposure [4]. These losses are caused by soiling on the surfaces of the heliostats.

2.4 Heliostat Cleaning

To counteract the efficiency losses brought about by soiling (dust gathering on the surface), heliostats are cleaned frequently, which is challenging on larger CSP plants. In the same study by Sandina Labs, they determined that merely spraying water at a pressure of 500 psi (3500 kPa) would restore mirrors back up to 95 percent reflectivity [4]. Currently the most common method of cleaning heliostats is high pressure water from a truck such as in Figure 2.3 [18]. The truck passes the heliostat while spraying water from an arm fitted with high pressure nozzles. To use the trucks, the ground near the heliostats requires preparation, and a large supply of water must be available. The trucks also carry the risk of damaging any equipment that they pass while cleaning

[19]. Although effective and widely used, the large water demand and risk associated with cleaning trucks leaves a need for more effective and compact cleaning solutions.

Figure 2.3: Typical heliostat cleaning truck

In recent years other cleaning methods have been created to solve the water demands and risks associated with cleaning trucks. The trucks have been fitted with brushes and chemicals have been added to minimize water usage. The focus for heliostat cleaning has shifted towards autonomous designs to be as efficient as possible. A robotic wiper was created by Taft Instruments [6], called the Valin One and was being tested in 2013. A wiper robot is fitted to each heliostat and uses electrostatic waves to clean the mirrors. Resin Engineering in Israel [20], [6] created an autonomous PV panel cleaner called the Solar Robot, which uses low levels of water to clean the panels. Although innovative, the Valin One will be costly due to the restriction of having one robot per heliostat, similarly, the Solar Cleaner is unable to move between heliostats which are not located close together.

The most promising cleaning robot currently exists as the HECTOR, developed by SENER [5]. The robot can position itself and navigate along the mirrors whilst achieving close to 100 percent reflectivity on the mirrors. The HECTOR is unable to navigate large gaps between heliostats and like the Solar Robot, cannot move itself from one set of mirrors to another. The HECTOR can be seen in Figure 2.4 [5].

2.5 Mirror Scratching

To maintain high levels of reflectivity, the cleaning process used on the surface of the mirror must ensure that the surface does not get damaged. It was found that for glass reflectors the cleaning medium can cause surface damage [21]. The study showed that contact cleaning on polymer film reflectors could yield a reflectivity drop of up to 8 percent over a period of 2 years. The reflectance drop occurred due to micro scratches created on the mirror surface. In the same

Figure 2.4: HECTOR cleaning robot by SENER to clean heliostats

study it was determined that mediums of contact cleaning with soft brushes (0.1 mm bristle thickness) did not cause scratching, thus maintaining high levels of reflectivity on polymer coated mirrors. The effects of hard contact cleaning were significant to the reflectance and the results can be seen in Figure 2.5 [21].

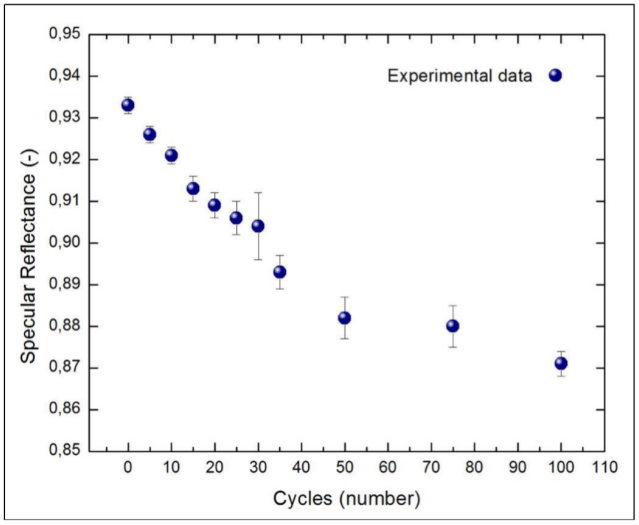


Figure 2.5: Reflectance vs number of cleaning cycles of hard contact cleaning methods

2.6 Robotics Operating System

The Robotics Operating System, or ROS, is an open-source platform that has been created for developing robots. It allows the user access to various tools and libraries to make programming robots more accessible, while being compatible with multiple programming languages to encourage collaborative robotic development worldwide [22]. ROS makes use of small programs known as nodes, these nodes are small programs created to perform a task. The

nodes send or receive data over a topic, which allows communication between the nodes. The method of using nodes means that preexisting nodes can be used in the development of a robot, and processes can be broken down into smaller, less complex components [23]. A basic node-topic layout can be seen in Figure 2.6 [24].

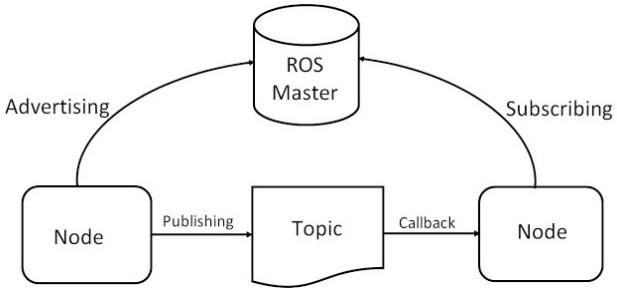


Figure 2.6: ROS nodes publish and subscribe to topics, which are used to send data between nodes. All ROS data is controlled by the master and nodes can publish and subscribe to multiple topics

ROS is also capable of low level device control and acts as an operating system for a robot, as it provides an underlying platform to run these nodes simultaneously [25]. Robots require sensors and sensor data, often in real-time. ROS thus allows sensors to obtain, process and analyze data so that the robot may achieve its tasks [26]. ROS has been developed with support for both the C++ and Python programming languages, but support for Java has been added into some libraries. The advantage of using nodes means that the data sent and received by nodes is irrelevant of the language it was coded in, allowing multiple languages to be present in one robotic system. ROS libraries with precompiled nodes are also readily available for many common robotic components, allowing for compatibility with most hardware [27]. The open source nature and choice of programming languages thus make ROS an ideal tool to develop a robotic system. Its ability to run on small computers such as the Raspberry Pi and Arduino is also advantageous.

2.7 Edge Detection

Edge detection is a form of image processing in which the edges of objects are extracted from an image. It uses varying brightness and colors to determine the possible existence of an edge. It is often used for gathering data in computer vision or robotic applications, with the most common methods of edge detection being Canny, Sobel, Prewitt, Roberts and Fuzzy Logic [28].

In OpenCV, the most common form of edge detection is Canny edge detection. It has been found that Canny edge detection out-performed all other tested methods although it has a higher computational cost [29]. It provided

the least amount of false edges while detecting more objects with feeble edges. The study also validated the fact that the computational cost increases with image resolution, regardless of the edge detection method used. Canny Edge Detection was created by John Canny and works in stages. First, noise reduction of the image is performed using a 5x5 Gaussian filter [30]. Gaussian filtering is a type of image blurring, which removes high frequencies such as noise and sharp edges from an image [31].

After filtering, the intensity gradient of the image is found. This is done using a Sobel kernel in the horizontal and vertical directions to get the first derivative in each direction. The Sobel kernel assigns a value to each pixel based on the pixel's intensity. By using the derivatives of these pixel intensities in the horizontal and vertical directions, the edge gradient can be found. The edge gradients, G, are computed using Equation 2.1 and the direction of the edge, γ is computed using Equation 2.2 [30]. G_x is the pixel intensity derivative in the horizontal direction, while G_y is the pixel intensity derivative in the vertical direction. The effects of applying Gaussian blur can be seen in Figure 2.7.

$$G = \sqrt{(G_x^2 + G_y^2)} \tag{2.1}$$

$$\gamma = \tan^{-1}(G_y/G_x) \tag{2.2}$$

The gradient direction is always perpendicular to the edges, and is rounded to the nearest vertical, horizontal, or diagonal axis. Once the magnitude of the gradients and their directions for each pixel are known, pixels which are not part of an edge are removed from the image. To do this, each pixel is checked if it is a local maximum in the direction of the gradient. if the pixel is found to be a local maximum, it is considered, it if is not, then it is set to a zero pixel [30]. After removing pixels, the image is left with only thin edges.

The final stage of edge detection makes use of a hysteresis filter to discard any values that fall outside a minimum and maximum threshold. The filter checks the intensity gradient of all the pixels thought to be lines and discards those that fall outside the threshold. The filter also checks which pixels are connected, to ensure that any pixels that may be isolated are also discarded. The filter leaves only strong edges in the image [30]. The resulting edge detection can be seen in Figure 2.7 [31].

2.8 Line Detection

Line detection is a form of image processing that extracts straight lines from an image. It often makes use of edge detection to find these lines. Line detection can be useful when extracting features from an image, as lines may provide information about absolute references points in a camera frame. The most common form of line detection is that of Hough lines, however, methods such

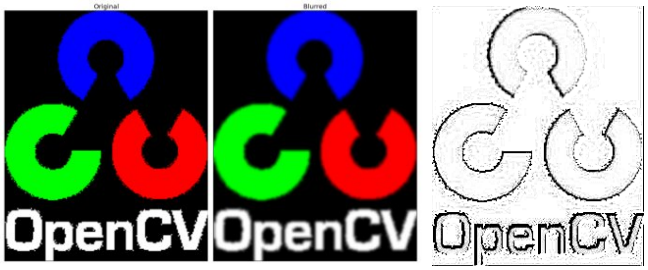


Figure 2.7: The effects of applying Gaussian blur and the resulting edges detected

as the Radon transforms also exist [32]. Hough line transform in particular, has been used in algorithms for horizon detection at sea, known as QHLD (Quick Horizon Line Detection) in a paper done by [33], as well as in mobile robot navigation systems in a study done by [34]. It is worthwhile to note that Hough lines is also used in cars for lane detection in self driving applications [35]. It was noted that Hough line detection can be slower than other conventional line detection algorithms, especially at higher resolutions, restricting its capability to work in real time, but provides greater accuracy over other methods [33].

Hough line detection was developed by Paul V.C. Hough to recognize complex lines in an image. To perform line detection, it is first assumed that a point in an image (x, y) can have any number of lines pass through it. Traditionally one represents a line using its gradient and intercept using the straight-line equation, the problem with this, however, is that lines that are vertical have infinite gradients and are problematic when used in computation. To solve this issue, Hough lines uses a line perpendicular to the line of interest, which passes through the origin. The line can be represented in polar co-ordinates and is given in Equation 2.3 [36].

$$r = x\cos(\theta) + y\sin(\theta) \tag{2.3}$$

The straight line is now represented in polar co-ordinates by a perpendicular line that passes through the origin as in Figure 2.8 [37], this allows for easier computation and prevents infinite gradients.

If one uses the assumption that at any given point, (x, y) there can be any number of lines that pass through it, and each line can be represented by Equation 2.3 with only $r > 0$ and $2\pi > \theta > 0$ considered [37], then one can graph all possible (r, θ) values at (x, y) as in Figure 2.9 [37]. These values form a sinusoid, and this is what is referred to as the Hough space [36].

The process is then repeated for all (x, y) points in the image where Canny Edge Detection has detected edges. The graphs for (r, θ) for all values of (x, y) are then all plotted on the same set of axes as in Figure 2.11 [37]. There are now multiple lines in the Hough Space. If any of these lines intersect in the Hough space it means that these different (x, y) points share the same straight line. Since edge detection has already been performed, one can assume these

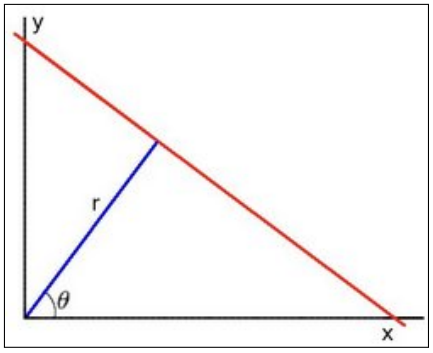


Figure 2.8: A line can be represented in both $x - y$ and $r - \theta$ space

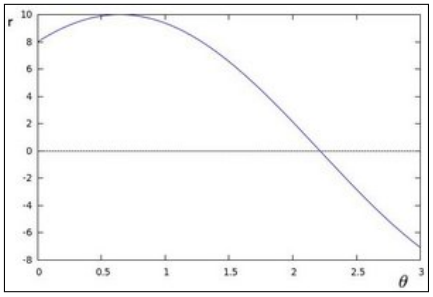


Figure 2.9: All possible lines that can pass through $(x = 8, y = 6)$, expressed in polar co-ordinates in the Hough space

points to be on the same edge. Hough line transform stores the number of intersections at each point in the image, and the Hough threshold can be used to set the minimum number of intersections needed for a line to be declared. In addition to the threshold, Hough lines can also be characterised by a minimum line length, which defines the minimum length in pixels for a line to be declared. A maximum line gap may also be specified, which ensures that all the points found on a line must be within a specified distance from one another to be declared part of the same line [37].

In addition to the threshold, line length and maximum line gap, there are the ρ and θ thresholds. When the line detection is done, and all the lines on the Hough space are plotted, the algorithm selects all the lines that match the Hough threshold, line length and gap criteria. However, sometimes the intercepts of these lines are not exactly in the same place on the Hough space, they may be one or two decimals away from one another for instance. The parameters for ρ and θ separate the Hough space into bins or rectangles, with sizes determined by the parameters. ρ determines the bin height and θ determines the bin width. This means that even though lines may not intersect at exactly the same place, if they fall into the same bin they can still be considered intersecting [37].

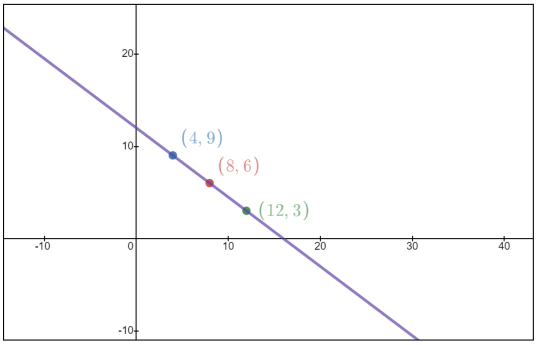


Figure 2.10: The three (x, y) values, $(x_1 = 4, y_1 = 9)$, $(x_2 = 12, y_2 = 3)$ and $(x_3 = 8, y_3 = 6)$, lie on the same straight line, meaning there will be an intersection for the straight line's (r, θ) value in the Hough space for all three (x, y) values

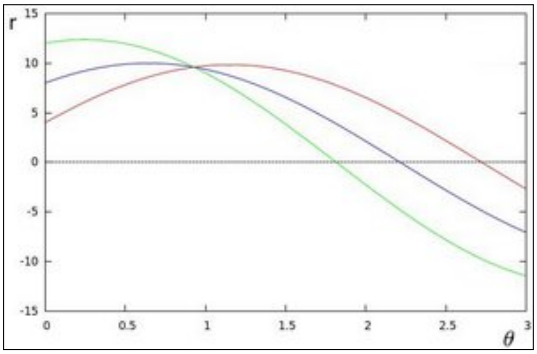


Figure 2.11: All possible lines that can pass through three (x, y) values, $(x_1 = 4, y_1 = 9)$, $(x_2 = 12, y_2 = 3)$ and $(x_3 = 8, y_3 = 6)$, expressed in polar co-ordinates in the Hough space. The point of indication suggests that there is a straight line that all three points lie on, which was shown in Figure 2.10

2.9 Localisation and Navigation of Robots

The localisation of a robot is the process of finding its position relative to its surroundings. This means that a robot should identify its own location based on what its sensors observe. Localisation is a critical component in autonomous robotics, as it allows a robot to make its own decisions based on the environment [38]. Localisation can be split into relative positioning and absolute positioning [39]. Relative positioning often makes use of encoders on the robot's wheels or drive-train to determine how far its moved from a reference point. Relative localisation can obtain errors due to wheel slipping or rough terrain. Absolute localisation makes use of sensors to determine the absolute position of the robot in an environment. Although more accurate, absolute localisation has a much higher computational demand over relative positioning, and often requires more advanced sensors [39].

Localisation has been implemented in different ways, with the most com-

mon forms being Extended Kalman Filter (EKF) in Landmark maps, Particle Filter (PF) for Grid maps and Simultaneous Localisation and Mapping. The Extended Kalman Filter and the Particle Filter both require a predefined map of the environment, which makes these methods impractical in areas that are unknown to the robot. The EKF requires a map of the landmarks in the environment, and the algorithm attempts to estimate a pose for the robot that most accurately reflects the movement and sensor data of the robot. EKF requires the robot to be able to associate its measurements relative to specific landmarks, otherwise failure can occur during navigation [38].

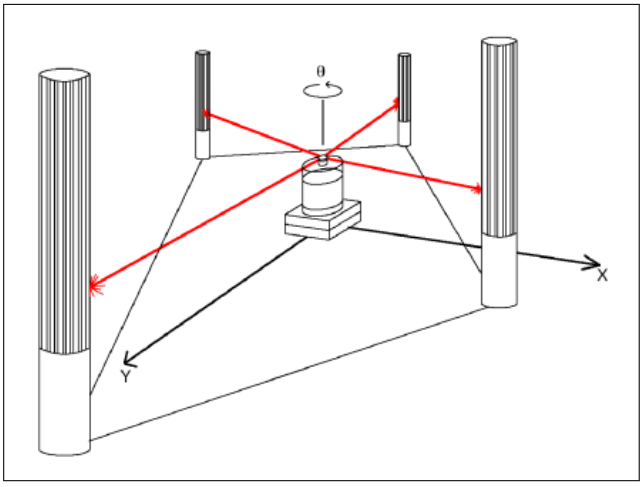


Figure 2.12: Absolute localisation of a robot using landmarks

When an environments map is available as a grid, or the initial localisation of the robot is unknown, an Extended Kalman Filter can no longer be used. A Particle Filter can be used in place of the EKF. A Particle Filter makes use of a segmented map, or grid, and estimates the pose of the robot based on its most likely position in the map. Weighted estimates are generated at each position, and as the robot moves through the map the weights are adjusted based on the sensor data and motion of the robot. Increasing the number of particles for the system does have a direct impact on the processing cost [38].

Alternate approaches to robot localisation include that of SLAM or absolute positioning (Figure 2.12 [40]) data among others. Simultaneous Localisation and Mapping (SLAM) is a combination of localisation and navigation where the robot generates a map of the environment and places itself within the map without prior knowledge of the surroundings. SLAM however, does require extensive computational requirements and is not always a practical solution to the navigation problem if a map of the environment can be created [39]. Absolute positioning can be used to localise a robot in the map. By using the robots sensors to obtain data about a known point in the environment, it can compare the position of this point with increasing time steps, allowing

information about its pose to be obtained [38]. A method that follows the idea of this absolute positioning was used to program the robot in this thesis.

Once a robot has localised itself within an environment, it needs to determine how navigate to its goal point. Path planning is a typical problem in the field of robotics, as it includes the ability of robots to avoid obstacles. Path planning can be broken up into 2D and 3D, and navigation can be separated into global and local navigation [41]. Local navigation uses the robot's sensors to obtain data about the immediate surroundings to get to intermediate goal points. Global navigation is most commonly implemented using the Artificial Potential Field (APF) or the Dijkstra method. The most common sensor for local navigation is LIDAR. LIDAR's are common in automation and can map their surroundings without the need for GPS. LIDAR is frequently used in SLAM applications [41].

The APF method assumes that all points in the map act like some form of magnetic field. Obstacles act as a repulsive force while the goal point is the attractive force. The magnitudes of the resulting attractive and repulsive forces are used to determine the direction in which the robot should move. The Dijkstra method searches the area in the map for all paths that lead to the goal point, and extracts the most optimal path to the goal point. The goal point for a robot does not need to be one fixed point, as the goal point can move once certain locations have been reached [41].

2.10 Sensors

For a robot to localise itself in any environment, it needs sensors [42]. Sensors come from one of two categories which are either interoceptive or exteroceptive, and sensor quality directly impacts its performance. Interoceptive sensors are used to obtain relative positioning, such as encoders. Exteroceptive sensors are used to give absolute measurements, such as lasers and cameras [39]. The most common forms of sensors are acoustic, laser and stereo vision sensors [42], however it is not uncommon for multiple sensors to be combined to obtain a better map of the environment [39]. The biggest challenge the heliostat cleaning robot faces is that of determining where the edges are, both when it is at the edge and when it is in the centre of the mirror.

2.10.1 Acoustic Sensors

Acoustic sensors are mostly comprised of sonar or ultrasonic sensors and use time of flight to measure distances by producing a cone of sound that is reflected by hard surfaces [39]. Sonar sensors are used underwater, where laser and visual sensors under perform. Ultrasonic sensors are relatively robust in terms of their overall cost but are easily influenced by noise [42]. Ultrasonic sensors are also incapable of differentiating objects by color or feature, and

since they rely on reflected sound waves they are only capable of seeing the closest object in their signal cone. Ultrasonic sensors are thus only capable of detecting the closest obstacle in their sound path, and further obstacles are invisible to the sensor. An ultrasonic sensor is shown in Figure 2.13 [43].

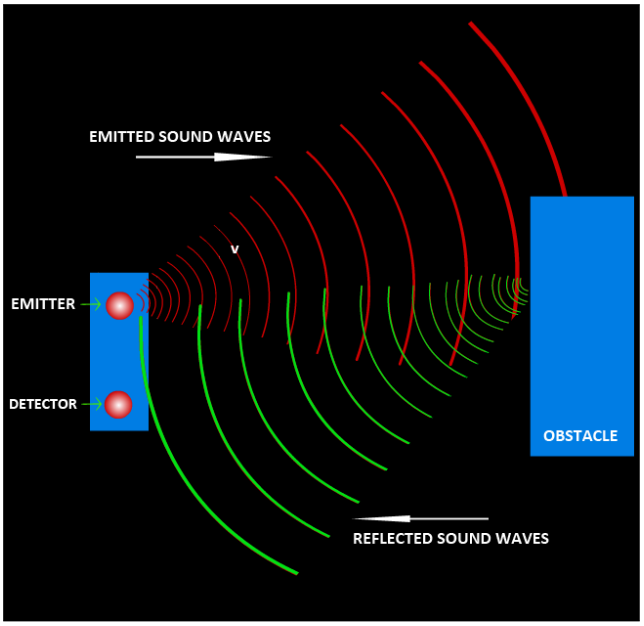


Figure 2.13: An ultrasonic sensor and its sound cone. The sensor is incapable of seeing objects past the first obstacle

2.10.2 Laser Sensors

Laser sensors are most common in SLAM applications and can provide accurate data in both indoor and outdoor environments. Laser sensors also use time of flight to measure distance but at a much faster rate, so their advantage over acoustic sensors is the speed at which they can obtain data, but they are also capable of detecting multiple obstacles in their field of view [39]. Laser sensors, or LiDAR are often positioned on a rotating motor, so that they may capture 360° around their position. The biggest disadvantage of laser sensors is their cost compared to acoustic sensors. Object identification is also difficult with laser sensors, as they cannot perceive colour differences in scanned features [39]. Laser sensors provide unexpected results when used on clear or reflective surfaces, and as such are not ideal around mirrors or windows [44].

2.10.3 Visual Sensors

Visual sensors are common in robotics and come in the forms of a monocular camera, stereo camera and RGB-d camera [39]. Monocular cameras consist

of a single camera that take 2D images, they have a low computational cost but require complex algorithms to solve positions in the image. Monocular cameras do not store any depth information, and as such data may be lost in the images. Stereo vision cameras use multiple 2D images to create a 3D scene but require more computational power than monocular cameras [39]. Stereo vision cameras are separated by a known distance, or baseline, and features such as corners or edges can easily be identified by both cameras, depth information can be extracted from the two images [45]. Stereo vision, as in Figure 2.14 [39], has been used in many robotic applications and can be combined with laser sensors for more accurate environment representations. Modern SLAM systems make use of RGB-D cameras, which use 3D images generated by time of flight and varying light levels. Although accurate, RGB-D sensors are limited in direct light, and are not reliable on highly reflective surfaces [39]. They are also costly with computational power, and are highly sensitive to changes in the light conditions [46].

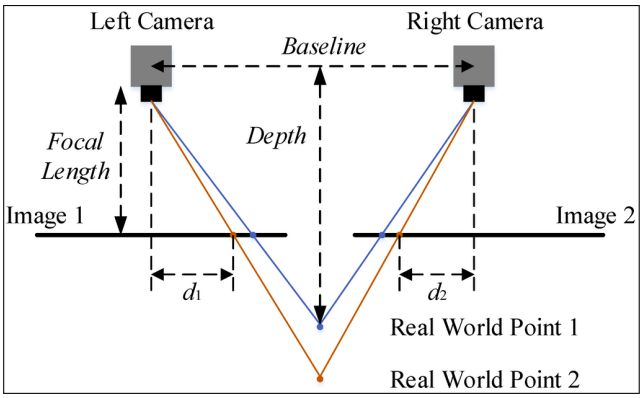


Figure 2.14: Stereo vision setup, two cameras are used to provide depth information from a scene

2.10.4 Other Sensors

In addition to the sensors mentioned above, magnetometers, GPS and IMUs are common sensors in the robotic industry. Magnetometers measure the magnetic flux around the robot, which have been used to determine the orientation of robots relative to the North Pole, however they are severely affected by magnetic fields. GPS or Global Positioning Systems use satellite triangulation to determine the position of the sensor, but their accuracy is usually within a ten-meter tolerance. IMUs or Inertial Measurement Units track the rotation and translation of a robot and can be used to determine how the robot has moved relative to a point.

2.11 Camera Model

For a robot to obtain its position from camera data about its surroundings, it needs a relationship between the 2D image plane and the 3D world. To obtain this relationship, the pinhole camera model can be assumed for any calibrated camera. A pinhole camera is created by placing a barrier with a small hole between a lens and the 3D scene. The barrier ensures that light emitted from other areas in the 3D scene are ignored by the camera lens. The only light that is able to reach the lens of the camera is that coming through the hole in the barrier, resulting in in an image of the 3D scene with a one-to-one mapping to the image [47]. The pinhole camera model thus projects 3D points onto the image plane using perspective transformation, allowing the 3D space in-front of the camera to be mapped to 2D points. The assumed pinhole camera model axis systems can be seen in Figure 2.15 [48].

Camera calibration is a means of estimating the intrinsic parameters of a camera. The intrinsic parameters of a camera include that of its focal length, distortion, and skew parameters, as well as the image centre of the camera. Intrinsic parameters are important when obtaining 3D data from a 2D image and are required when using the pinhole camera model assumption. Camera calibration is done by showing the camera a pattern of known size and geometry (often a checkerboard, see Figure 2.16) at different poses and positions. The camera then processes these poses, knowing the size of the pattern, to provide the cameras intrinsic parameters. These parameters are important when analysing the position of objects in a camera scene.

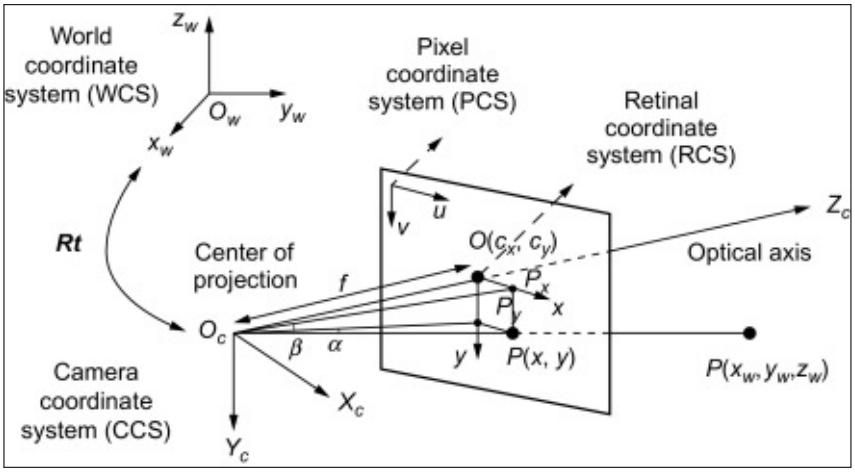


Figure 2.15: Pinhole camera model axis systems. A point, P, in the image space can be mapped to the world space using the pinhole camera model

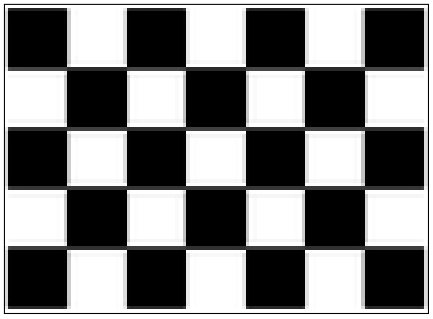


Figure 2.16: Checkerboard used in camera calibration to determine the intrinsic parameters of a camera

In Figure 2.15 the optical centre, O_c is where the camera's lens is located. The image co-ordinate system, (u, v) or O, is fixed to the image plane, and the world co-ordinate system, (X_w, Y_w, Z_w) or O_w, can be anywhere in the world space. One can obtain the world co-ordinate of a point if one has its image co-ordinates and vice-versa using the pinhole camera matrix. The relationship from the image co-ordinate to the camera lens co-ordinate is given in Equation 2.4, which is a transformation matrix from the image plane to the camera lens centre based on the intrinsic parameters of the camera (focal lengths and lens distortion). In Equation 2.4, u and v are the image co-ordinates, X_c, Y_c and Z_c are the camera co-ordinates, f can be separated into f_u and f_v, which are the focal lengths in the u and v directions, and C_x, C_y are the centre co-ordinates of the image plane. Z_c is a scaling factor to account for changes in depth of the image [49].

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & C_x & 0 \\ 0 & f_v & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.4}$$

Similarly, the world co-ordinate system, (X_w, Y_w, Z_w), can be mapped to the camera co-ordinate system, (X_c, Y_c, Z_c) with Equation 2.5. In Equation 2.5, R is the rotation matrix between the camera co-ordinates and the world co-ordinates, and t is the translation matrix between the camera co-ordinates and the world co-ordinates. [49]

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.5}$$

If Equations 2.4 and 2.5 are combined, the resulting Equation 2.6 is obtained. The values $r11$ through $r33$, and $t1$ through $t3$ are the rotation and translation values between the image and world frame. The equation

can be used to map camera image co-ordinates, (u, v), to world co-ordinates, (X_w, Y_w, Z_w) [49].

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & C_x \\ 0 & f_v & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.6}$$

Chapter 3

Design

The design section discusses all the requirements, specifications, concepts, and decisions made to the design the proof-of-concept with the capability to localise and navigate itself over a heliostat. Justification for all decisions made are discussed where necessary. The robot was designed to navigate various sizes of heliostats, while using a camera as the primary means of detecting the heliostat edges. Larger heliostats (Figure 3.1 [50]) can have gaps between the mirror panels which was also considered for the design. The robot was also designed to move between freestanding heliostats, using a drone, but this was not tested at the time of writing.

Figure 3.1: A large heliostat with multiple small mirrors. These mirror are separated by small gaps which the robot will need to navigate

3.1 Functions and Requirements

This section provides both the design requirements and functions of the system. Each requirement was linked to its respective hardware component, to ensure that each one was accounted for. Table 3.1 provides all these requirements and allocated them to a hardware component. The requirements were developed to ensure that the robot was capable of successfully navigating the heliostat for the purpose of cleaning.

Table 3.1: The robot design requirements and the respective hardware components needed to achieve them

	Actuators	Sensors	Embedded PC	Power Supply	Pump	Cleaner	Water Supply
Clean heliostats autonomously	x	x	x	x	x	x	x
Clean various shapes and sizes of heliostat	x	x	x	x	x	x	x
Navigate heliostat safely	x	x	x	x			
Communicate wirelessly to send and receive data			x				
Provide water to cleaning tool					x		
Have a small turning radius	x						
Provide controlled speed to the wheels	x						
Connect with all sensors and actuators			x	x			
Process data in real time			x				
Use ROS on the robot			x				
Detect or recognise heliostat edges		x	x				
Determine robot's orientation relative to heliostat edges		x	x				
Power all hardware on the robot				x			

3.2 Specifications

The system specifications use the requirements to allocate achievable quantities to each requirement, this allows one to benchmark the system to determine whether the requirements were achieved or not. Assigning a value to the requirement helps to determine if they have been achieved or not. Some requirements are not allocated quantities as they are not able to be quantified. Table 3.2 provides the specifications.

Table 3.2: Design specifications for the robot, derived from the requirements

Specification	Value	Unit	Motivation
Maximum Cleaning Time	5	minutes/m^2	The robot needs to be quick enough to clean many heliostats
Minimum Heliostat Size	2	m^2	Heliostats under this size are uncommon
Maximum Edge Identification Time[1]	10	seconds	The robot must identify the lines quickly, to ensure quick cleaning
Minimum Battery Life	2	hours	Longer battery life was desired, as short life wont allow for much cleaning
Maximum Mass[2]	4	kg	For future development, the robot will be mounted to a drone with this lifting capacity
Minimum Water Supply Size	500	ml	This water capacity should be sufficient to clean multiple mirrors
Maximum Time Between Sensor Data	200	milliseconds	Data needs to be gathered in real time to prevent navigation failure
Maximum Orientation Error[3]	5	°	If the robot orientation error was too large, the robot may not navigate the whole mirror surface

[1]The line identification time specifies the maximum amount of time the robot must use to find an edge once being placed on the heliostat.
[2]The maximum mass was determined by the maximum lifting capacity of a drone available at the time of the project.
[3]To keep the robot movement precise on the mirror, the error for its orientation should be as little as possible.

3.3 Concept Generation

Concept generation is a natural step in the design of any prototype. Concepts for the robot design are analysed to ensure it meets the set-out requirements and specifications. The concepts are separated into three sections, robot concepts, wheel concepts and sensor concepts. The robot concepts discuss the robot's operation in general, while the wheel and sensor concepts discuss the hardware needed to navigate them. Concepts in the robots cleaning pattern are also discussed in this section. Additional concepts which were not considered here may be viewed in Appendix A.

3.3.1 Robot Concepts

The first concept was a heliostat wiper similar to the Solar Robot [20]. The robot attaches to the longest edges of the mirror while rollers are used to move across its surface. This allows for easy control, as distance sensors can be used to detect the short mirror edges. However, moving the robot between heliostats would be challenging due to the size and weight. It was also restricted to the size and shape of heliostat that it was capable of cleaning, as the concept was designed to fit one mirror. The concept may be seen in Figure 3.2.

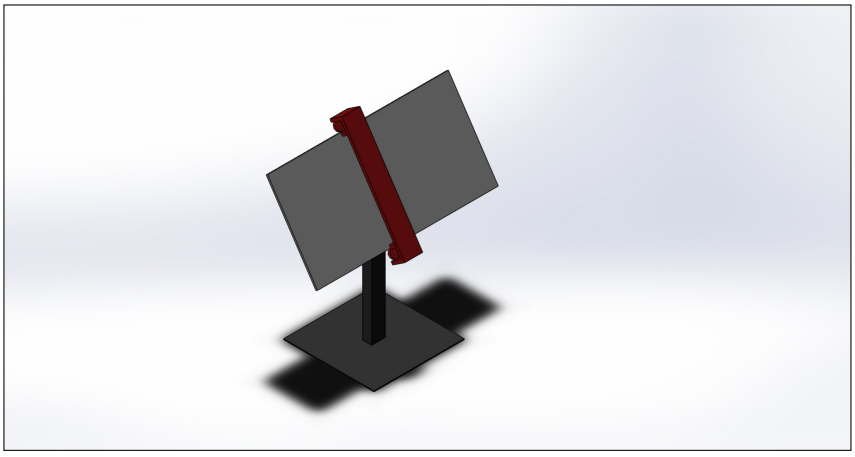


Figure 3.2: Concept 1 - a wiper robot which fixes itself to the long edges of the heliostat, while moving from left to right to clean the surface

The second concept was a cleaning robot fixed to a drone. The drone lands on the mirror, powers down and the robot then cleans its surface using its own actuators and power supply. Localisation of this robot on the mirror will be challenging, but the system can clean all heliostat shapes and sizes. The drone allows easy movement between heliostats, and automation of the system was possible. The concept may be seen in Figure 3.3.

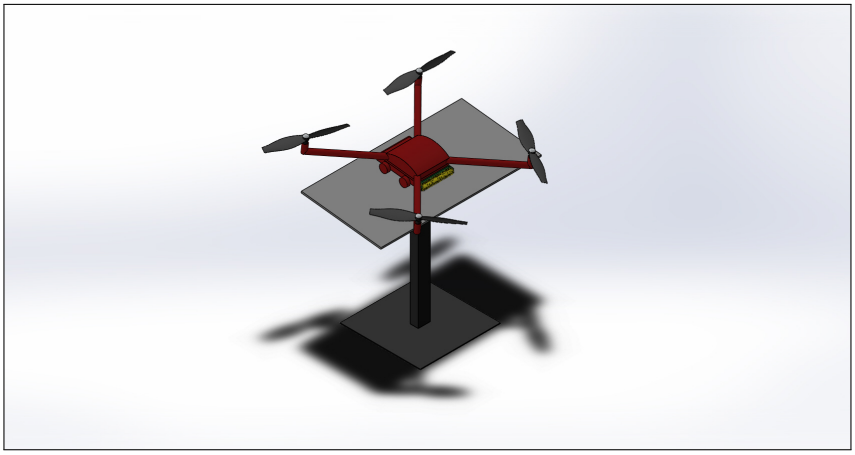


Figure 3.3: Concept 2 - a robot fitted to a drone which lands on the heliostat and navigates its surface

The third concept attaches to all sides of the mirror, but unlike the first concept, the cleaning section of the robot was smaller. The robot can move in both directions of the mirror plane using sliders, while being as light as possible. This would be simple to implement, but the robot's footprint would be as large as the heliostat and will prove challenging to move between heliostats. Scaling this robot to clean larger or smaller mirrors was challenging, and heliostats with a curved mirror may not clean as effectively. The concept may be seen in Figure 3.4.

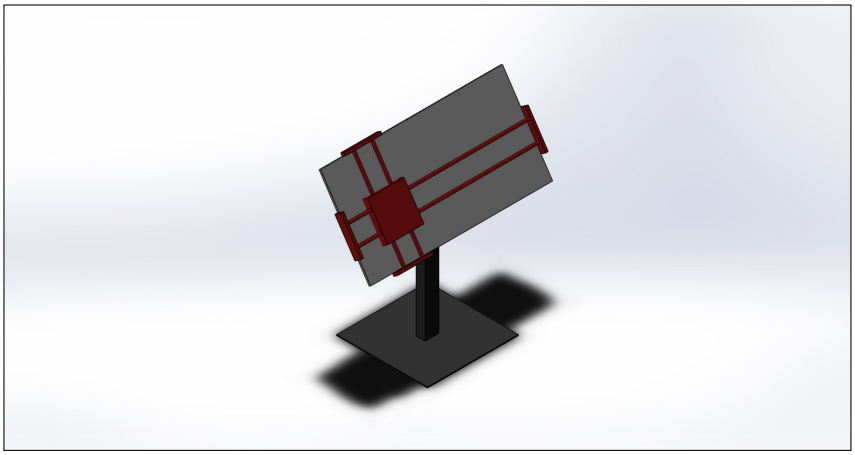


Figure 3.4: Concept 3 - a small wiper robot which fixes itself to all edges of the heliostat

The fourth concept also uses a drone, but the cleaner and its movement are controlled by the flight of the drone. This system would likely be quick to clean, but power consumption will be significant. Navigation around the mirror will be difficult if the drone was always in flight. This system however

would be easily scalable to fit many heliostat sizes and was capable of moving between heliostats. The concept may be seen in Figure 3.5.

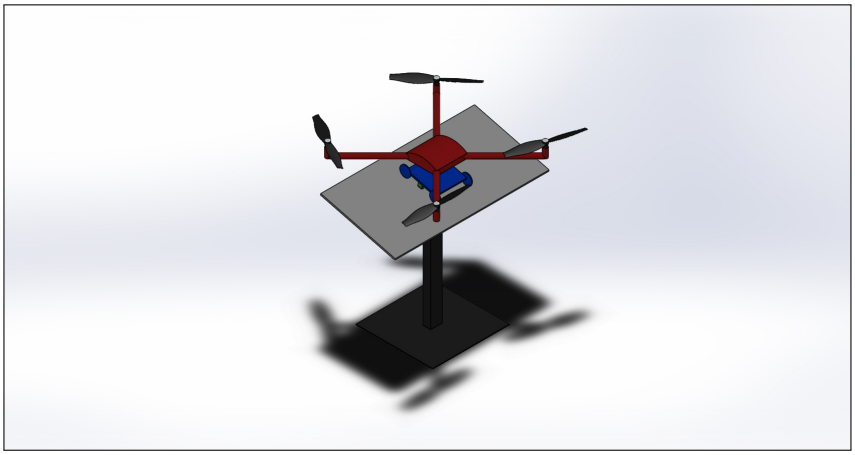


Figure 3.5: Concept 4 - a cleaner drone that flies over the heliostat surface while cleaning

With various robot concepts considered, concept selection was done with a scoring table. The scoring table scores each of the concepts according to different criteria. The maximum score for each cell was ten, being the best possible score, with the minimum score being zero, being the lowest score. Each criterion was weighted equally. The scores can be seen in Table 3.3.

Table 3.3: Main concept evaluation table

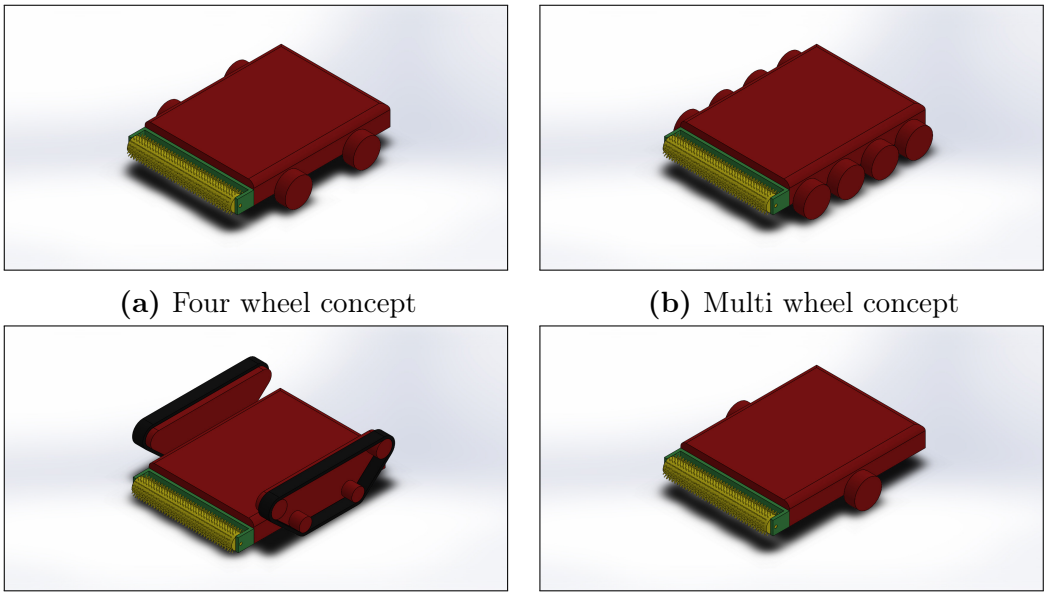
Concept Number:	**1**	**2**	**3**	**4**
Battery life - how long the system was expected to last	8	4	8	2
Adaptability - potential to clean different heliostats	2	8	2	8
Navigation - how simply the robot can navigate a mirror	10	6	10	4
Practicality - how likely the system will succeed	4	8	4	6
Robustness - how well the system can avoid problems	3	7	3	7
Maintainability - how easy the system was to maintain	6	5	2	5
Reliability - the possible consistency of the robots performance	10	7	9	6
Industry potential - how well the design can be commercialised	3	7	3	6
Growth potential - how much the system can be improved	3	8	3	7
Total score	49	60	44	51

Table 3.3, also shows that the the concept of a cleaner mounted to a drone achieved the highest overall score. This system makes use of a robot fitted to a drone, which lands on the heliostats. Once landed on the heliostat, the drone

will power down and the robot will navigate the mirror with its own actuators. This was used for the rest of the design as well as the consideration for the rest of the concepts to follow.

3.3.2 Wheel Concepts

Cleaning the heliostat requires the robot to navigate once on the mirror, to do this the robot requires wheels. The first wheel concept uses four or more wheels, each with its own motor. By powering each motor, it was capable of small turns, but may cause the wheels to slide on the surface when turning leaving scratches. Alternatively, one wheel on each side can be powered by a motor, while the others can rotate freely. This concept can be seen in Figure 3.6a, with four wheels, or in Figure 3.6b with six wheels.
The next wheel concept uses treads. Treads would easily clear any gaps but may also cause scratching on the surface of the mirror due to sliding. Keeping the treads tensioned could also prove difficult for the system. This concept can be seen in Figure 3.6c.
The last wheel concept in Figure 3.6d only uses two wheels. The cleaning tool was used as the other point of contact on the mirror. Having two wheels allows the robot to perform tight turns while minimising the sliding or scratching on the mirror.



(a) Four wheel concept

(b) Multi wheel concept

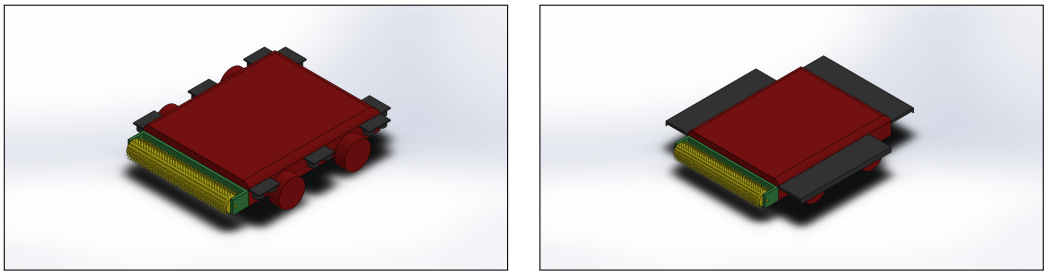
(c) Tread wheel concept

(d) Two wheel concept

Figure 3.6: Wheel concepts for the robot. The robot can be driven by four wheels or more. Treads may also be used and will navigate heliostat gaps. Two wheels can also be used with the last point of contact being the cleaning tool

3.3.3 Sensor Concepts

Sensors are critical in ensuring that the robot can navigate the mirror. The robot will need to identify when it has reached the edges of the mirror and have some form of absolute reference to the edges when in the middle of the mirrors. Sensor concepts are considered to select the best sensors for the system.
The first concept uses either ultrasonic or infrared light. The sensors face down at the mirror to detect when the robot has reached the mirror edge. The ultrasonic sensor will read high values when over the edge, and the infrared sensors will receive reflected light when over the mirror. Ultrasonic sensors are known to be relatively inaccurate, however this application was not typical to that of ultrasonic sensors, and accuracy was not too critical in detecting the edges. Infrared light sensors are more accurate, but light from the sun may cause interference in the sensor if the sunlight was able to get into the light receiver. The ultrasonic sensor concept can be seen in Figure 3.7a and the infrared can be seen in 3.7b.

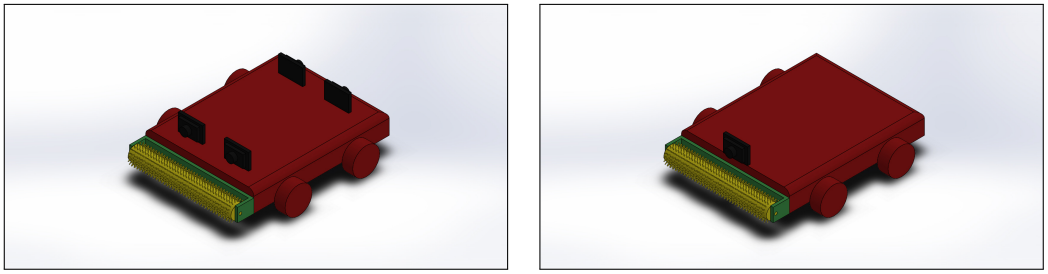


(a) Ultrasonic sensor concept **(b)** Light sensor concept

Figure 3.7: Sensor concepts for edge detection. The downward facing ultrasonic sensors are used to detect the heliostat edges. The light sensor reflects light off the mirror surface, if the light was not returned an edge was detected

The next set of sensor concepts was cameras. The first concept in Figure 3.8a makes use of two cameras for stereo vision to determine the position of the mirror edges from the robot. Stereo vision is costly in terms of computational power, but two images can provide more data and depth than a single image. The second camera concept in Figure 3.8 uses only one camera. The forward-facing camera will need sufficiently less computational power but lacks the depth information two cameras can provide. Line detection processes such as Hough Lines can be used to detect the position of the robot relative to the edge.

The last set of sensor concepts attempts to obtain data all around the robot. The first concept, in Figure 3.9a would use a camera with a wide field of view. This allows the robot to see all the edges of the heliostat at the same time, but these cameras are not easily accessible, and the computational requirement of a high field of view may be significant. The last concept aims a camera at a



(a) Stereo vision concept **(b)** Monocular camera concept

Figure 3.8: Camera concepts for the robot. Stereo vision uses two cameras to give the robot a better perspective of the environment. A monocular camera can be used along with the pinhole camera model to provide information about the edges

conical mirror to allow the camera to see the entire mirror without needing special lenses or fields of view. It may be difficult to obtain a mirror of this shape however, and the weight of the mirror could be significant. This can be seen in Figure 3.9b.

(a) Hemispherical camera concept **(b)** Conic mirror concept

Figure 3.9: Wide angle sensor concepts. The hemispherical camera was able to see all directions around the robot. The conic mirror aims a camera up at the mirror, and the reflected image shows the mirror area around the robot

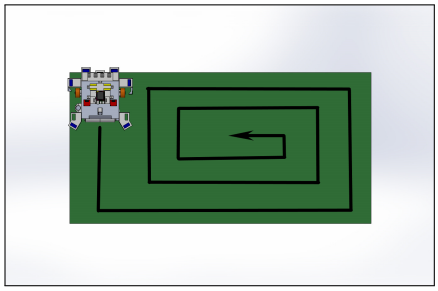
3.3.4 Selected Wheel and Sensor Concepts

After careful consideration of all the wheel and sensor concepts, it was decided that the robot would use two wheels, with the cleaner being the last contact point on the mirror. This was chosen due to it being least likely to scratch the mirror, and that it only requires two motors to be extremely manoeuvrable with a small turning radius. Downward facing ultrasonic sensors in combination with a forward-facing camera are chosen for the robot. Ultrasonic was chosen over infrared as it was likely that the sun would affect infrared readings, and accuracy was not of concern when finding the edges. The single camera

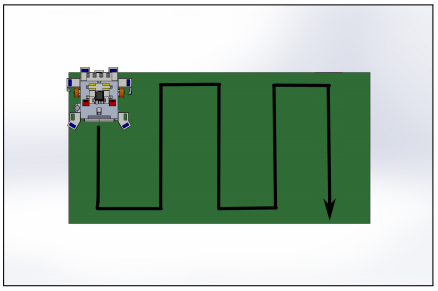
was chosen due to its computational efficiency, and when combined with computer vision algorithms, was expected to be sufficient in finding the position of the robot relative to the edges. Encoders are also considered to provide data on the robot's movement.

3.3.5 Cleaning Patterns

There are multiple patterns in which the robot could navigate the heliostat for cleaning. The first approach was to move in a spiral, such as in Figure 3.10a. The robot would first navigate the outside edge of the mirror, and work its way inward, ending at the centre of the heliostat. The next approach was moving between the edges of the mirror, such as in Figure 3.10b. The robot would start in one corner and drive from edge to edge, moving further over after each pass. The last approach would be to let the robot navigate freely and clean while doing so, however this would be significantly slower than the previously mentioned methods. The selected cleaning concept was the approach in Figure 3.10b in which the robot navigates between the edges and was discussed in the detailed design.



(a) Spiral cleaning pattern. The robot navigates the outside edges slowly making its way to the center of the heliostat



(b) Straight cleaning pattern. The robot navigates between the longest edges until it reaches the opposite side of the heliostat

Figure 3.10: Possible cleaning patterns for the robot

3.4 Detailed Design

The detailed design discusses the components that were selected for the proof-of-concept robot, as well as the navigation algorithm and the structure used in ROS to program the robot. Pseudo code is also provided in Appendix B to support the navigation algorithm. The design considers all the requirements established in section 3.1.

3.4.1 Component Selection

With concepts generated, components needed to be selected. Each component selected for the robot was discussed here to provide an understanding for the selection. A few components were not available at the time of writing, but alternatives were used in their place.

Embedded Computer

The embedded computer was chosen as a Raspberry Pi 3 B+ due to its capability of running ROS, and ease of programming in the python language. Although it is not as user friendly as an arduino when using motors and ultrasonic sensors, its ability to use cameras and opencv in python was the deciding factor for the writer.

Ultrasonic Sensors

Six HC-SR04 ultrasonic sensors were chosen, so that two could be placed on the front, two on the sides, and two on the back. The sensors have an operating distance of between two centimeters and four meters. These ultrasonic sensors are chosen due to them being low cost but robust. They are also easy to connect to a Raspberry Pi, although an arduino is usually recommended as a hardware controller.

Camera

The selected camera was a RaspiCam V2 with the standard lens. The camera has a maximum sensor resolution of 3280x2464 but are limited by computational power at this resolution. The camera was fitted at an angle of 10° from the vertical plane on the front edge of the robot. This camera was selected as it was easily connected to a Raspberry Pi.

Motors

Two 12 V 99:1 motors are used for the wheels, each providing a stall torque of 18 newton meters. At the time of writing, the availability of motors was scarce, and 6 V or 12 V motors were the only options available. It was decided that a motor with a higher torque would be beneficial to the robot due to the weight of the robot and the drone, and as a result 12 V was selected over 6 V. 12 V power supplies were also easier to obtain at the time of writing.

Cleaning Tool

A 50 mm diameter roller was used as the cleaning tool for the proof-of-concept and was made to turn with its own 12 V motor. The roller was covered in soft

cloth to prevent scratching on the mirror. The motor was also chosen as 12 V as it was capable of connecting to the same supply as the two wheel motors mentioned previously.

Wheels

Two 63mm diameter wheels are chosen for the motors. The wheels have a soft rubber tread and were the largest available diameter in supply at the time of writing.

Encoders

Magnetic encoders are fitted to each of these wheels, with a resolution of 8 state changes per wheel rotation. Ideally, motors with built in encoders should be used, but these were not available at the time of writing.

Water Pump

A 12 V peristaltic pump was fitted to the robot to provide water to the cleaning roller from a small water supply with a capacity of 500 ml. As mentioned with the 12 V cleaning motor, the pump was chosen to be 12 V to use the same supply as the motors. The peristaltic pump ensures that water was kept separate from any electronics.

Batteries

An 11.1 V, 3S LiPo battery was fitted to the robot to power both the 12 V motors for the wheels, the 12 V pump and 12 V cleaner motor. Although no batteries at 12 V were available at the time of writing, the 11.1 V battery was sufficient to power all the actuators. A 3.7 V LiPo battery was also fitted to power the embedded PC, the ultrasonic sensors and encoders.

Motor Drivers

Two L298 dual motor drivers are fitted to the robot to drive the wheel motors, the pump and cleaner motor. These motor drivers are chosen as they support 12 V motors and are capable of being driven from the Raspberry Pi GPIO pins.

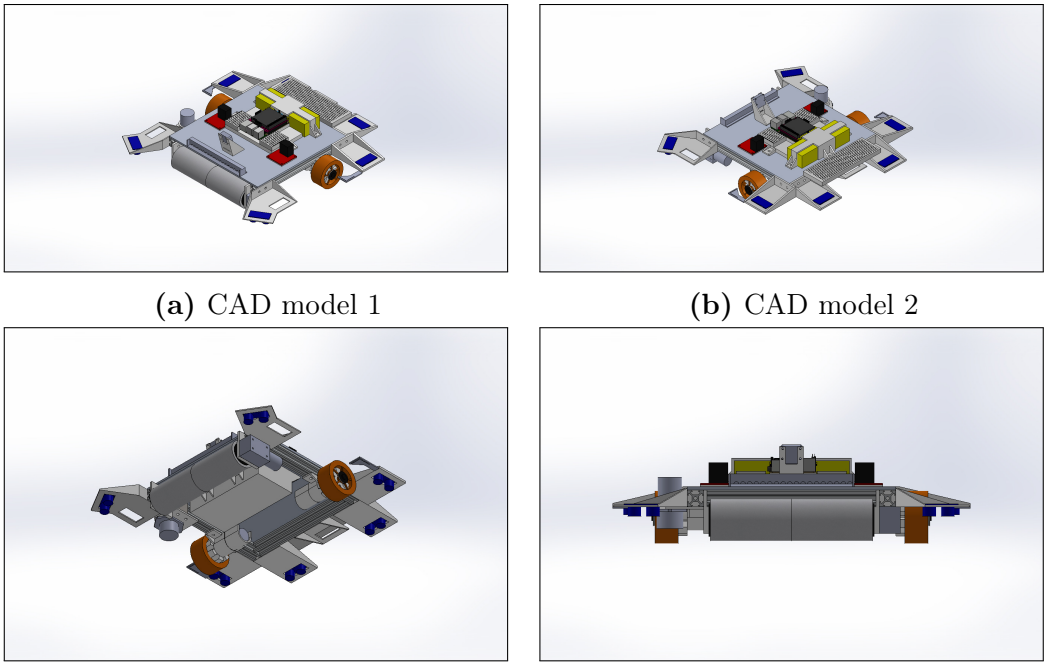
Other

All mountings for the robot and its sensors are 3D printed using PETG filament. PETG is durable and suitable for outdoor use, where other filaments such as PLA or AB warp when exposed to direct sunlight. The base of the

robot was assembled using 20mm x 20mm slotted aluminium, and the base plate was aluminium sheeting. Breadboard wires are used to connect all the electronics components together where soldering was not possible.

3.4.2 CAD Model

The robot was modelled using SOLIDWORKS so that parts for the build could be 3D printed. The CAD Model can be seen in Figure 3.11. Six ultrasonic sensors are fitted around the perimeter of the robot with the camera facing forward. The wheels are found behind the centre of mass to ensure that the cleaning roller has as much downward force as possible. The water tank was stored on the bottom of the robot and the CAD estimated the weight of the system to be 3.2kg without water. The robot was designed to be mounted to the underside of a drone.



(a) CAD model 1 **(b)** CAD model 2

(c) CAD model 3 **(d)** CAD model 4

Figure 3.11: Robot CAD model. The robot was first designed in CAD to ensure the right parts could be 3D printed

3.4.3 Final Build

The final build with all the components is seen in Figure 3.12. The final weight of the robot was 2.9 kg without water, and 3.4 kg with water. The build accurately reflects the CAD model in shape and size.

(a) Final build 1 **(b)** Final build 2

Figure 3.12: Final build of the proof-of-concept robot

3.4.4 Sensors

To further understand the sensor selection, it is important to understand the challenges the robot will face when navigating the heliostat and detecting the edges. The robot needs to be aware of any edges, both when it was near them and when it was in the centre of the mirror. Ultrasonic sensors are chosen to detect when the robot was at the edges as they are relatively low cost and provide a reliable means of detecting distance. They are not affected by any changes in light and make minimal use of the CPU. Since ultrasonic sensors only provide data near the edges of the mirror, additional sensors are required. The encoders are chosen as they track the robot's movement, but errors are likely in encoders especially with such low resolution, to get an absolute reference of the robot's position on the mirror, a camera was also used. The camera allows the robot to identify the edges of the mirror when the robot was not near any of the heliostat edges. The robot has enough sensors for identifying when it was near the edges, as well as where it was relative to an edge when in the middle of the mirror. Sensors such as GPS and an IMU were omitted from the proof-of-concept, as it was determined that the robot would not need them to navigate the heliostats. A magnetometer would likely be affected by the metal in the heliostats and was thus left out of the proof-of-concept.

3.4.5 Actuators

To navigate the mirror, actuators are required which make use of the sensor data to drive the robot. The wheels are each fitted to a 12 V DC motor which allow for accurate and controlled motion. The position of the motors allows the robot to make tight turns with minimal wheel contact on the mirror. The cleaner roller was connected to a motor which allows the roller to turn, acting as a third wheel for the robot. A pump was also used to move water from the storage tank to the cleaning roller, keeping the roller moisturised while the robot navigates the mirror. The positions of all the ultrasonic sensors, motors and pump can be seen in Figure 3.13.

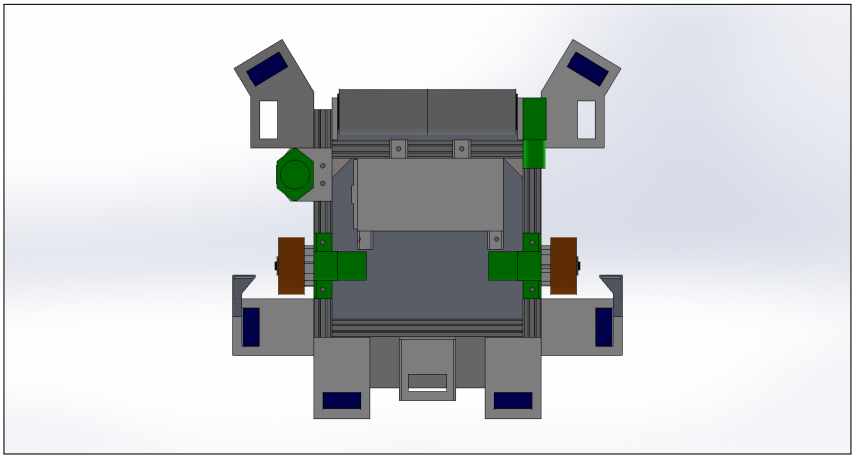


Figure 3.13: Underside of robot with the position of the actuators highlighted in green

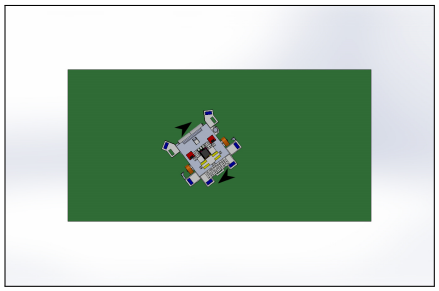
3.4.6 Navigation Algorithm

To begin navigating the heliostat, the robot must find one of the long edges of the mirror, this edge was used as the initial reference point. State 1 involves finding this edge, to do this the robot rotates on the spot until the camera detects the edge. Once found, the robot drives towards it in state 2. It will drive towards the edge until the front ultrasonic sensors have detected the edge. When at the edge, it uses both the front ultrasonic sensors to square itself up to the edge. The robot will then make a ninety degree turn at the edge for state 3, and drive along the edge in state 4 until it reaches the corner. Once the corner was reached, it will perform another ninety degree turn in state 5 and the cleaning process begins from this corner. The robot follows the first edge using the ultrasonic sensors until it has completed the whole short edge. State 7 was now reached. States 1-6 can be seen in Figure 3.14, and states 7-12 follow on the next page.

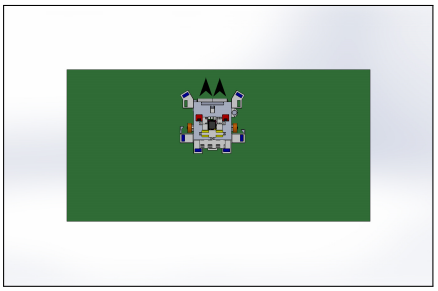
State 7 rotates the robot one hundred and eighty degrees left at the second corner. The rotation moves the robot to the right, so that the robot was not cleaning the same line again, however, it does overlap the previous path to ensure all the surface area was covered. State 8 takes the robot straight across the mirror to the opposite edge, using the camera and encoders, where it will then reach state 9, another state that turns the robot around. This process was then repeated, until the robot has reached the far edge, at which point it will end in state 11 or 12 depending on how many turns the robot needs to make. The cleaning process was now complete. States 7-12 can be seen in Figure 3.15.

It was important to note that each time the robot reaches an edge at the front, the ultrasonic sensors are used to square the robot up to that edge. This process was also done on the back edge, after the robot performs one hundred and eighty degree turns. When in the middle area of the mirror, the camera

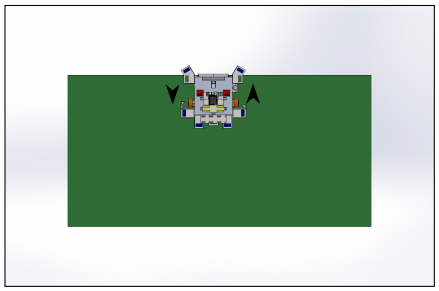
was used to detect the orientation of the edge in front of the robot, if the edge was not seen by the camera, a proportional controller was used with the wheel encoders to ensure that the robot was driving straight across the mirror. A different algorithm was used for driving along the edges, which uses the two ultrasonic sensors on each side to stay as close to the edge as possible.



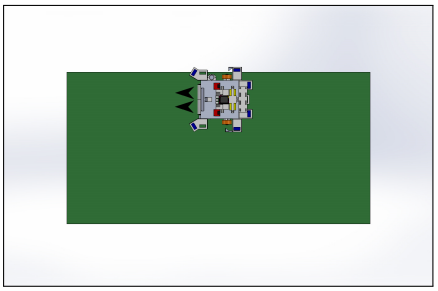
(a) Cleaning state 1 - The robot rotates on the spot, searching for the heliostat edge with the camera



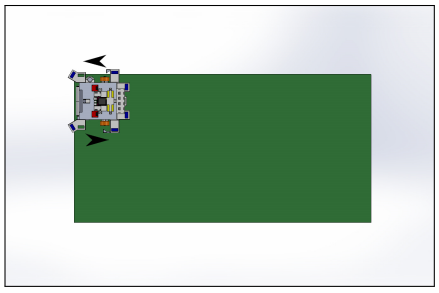
(b) Cleaning state 2 - Once the edge was found, the robot drives towards it until the ultrasonic sensors detect it



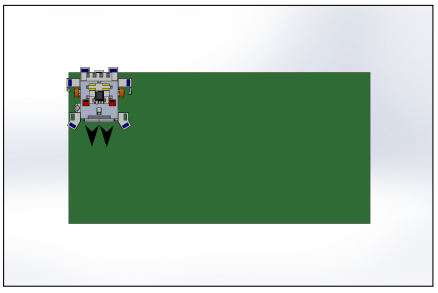
(c) Cleaning state 3 - The robot rotates left by ninety degrees



(d) Cleaning state 4 - The robot drives until it reaches the corner where the ultrasonic sensors detect the edge

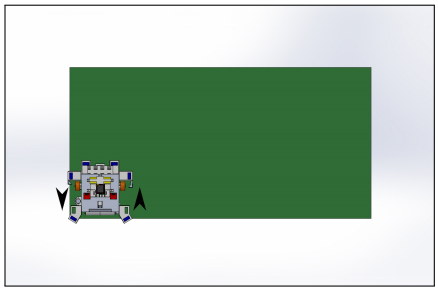


(e) Cleaning state 5 - The robot rotates left by ninety degrees

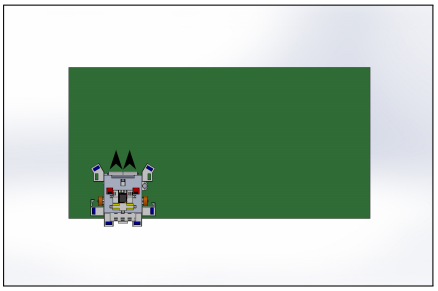


(f) Cleaning state 6 - The robot drives along the short edge until the ultrasonic sensors detect the opposite corner

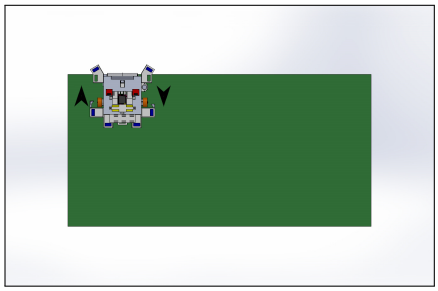
Figure 3.14: Cleaning states 1-6. Each state covers small movements of the robot, and state 6 ends when the robot reaches the opposite corner



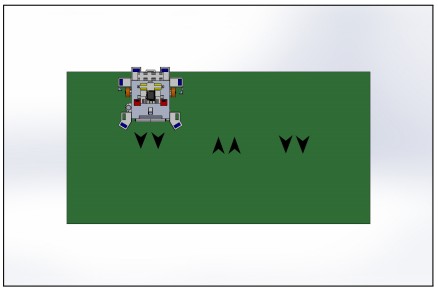
(a) Cleaning state 7 - The robot rotates one hundred and eighty degrees to the left



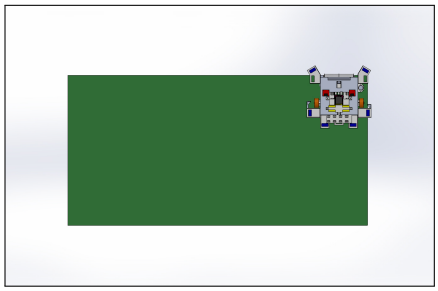
(b) Cleaning state 8 - The robot drives across the heliostat until it reaches the other end when the ultrasonic sensors detect the edge



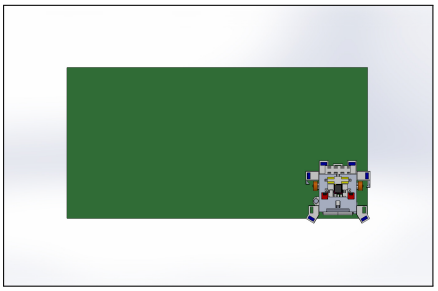
(c) Cleaning state 9 - The robot rotates one hundred and eighty degrees to the right



(d) Cleaning state 10 - The robot drives across the heliostat until it reaches the other end when the ultrasonic sensors detect the edge



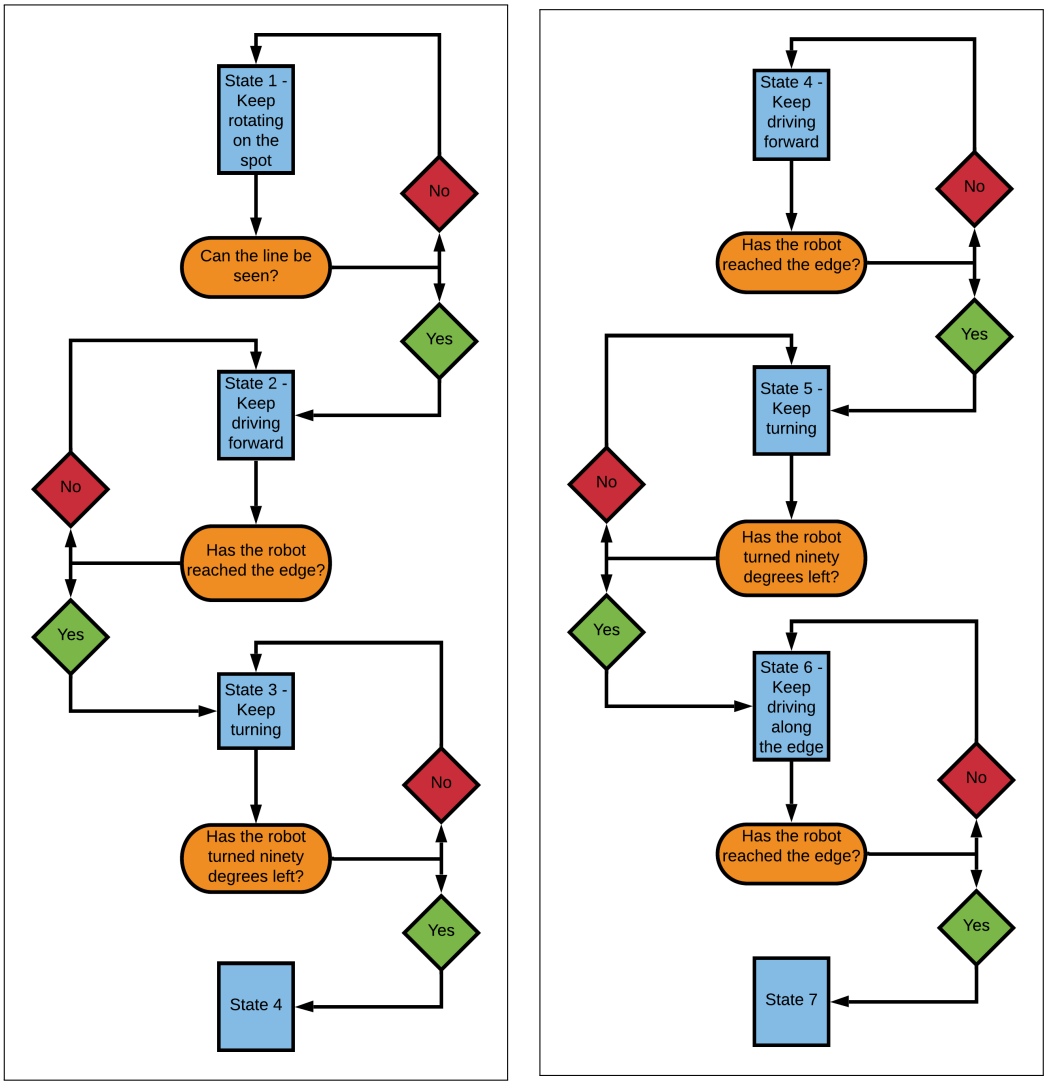
(e) Cleaning state 11 - The robot reaches the last edge on its right side, so it uses the right ultrasonic sensors to follow the last edge



(f) Cleaning state 12 - The robot reaches the last edge on its left side, so it uses the left ultrasonic sensors to follow the last edge

Figure 3.15: Cleaning states 7-12. The robot has two possible end states, 11 and 12. The state in which the robot ends with was determined by the length of the mirror. The robot was capable of determining its ending state automatically

Flow charts are provided to further explain the navigation process. Figures 3.16 to B.1 provide the logic to the process. States 1-7 are displayed in Figure 3.16, states 7-12 are displayed in 3.17 and states 11 and 12 are shown in more detail in the Appendix B Figure B.1.



(a) Flow chart between states 1-4

(b) Flow chart between states 4-7

Figure 3.16: Flow chart of states 1-7 for the robot. The chart highlights the conditions required to move to the next state

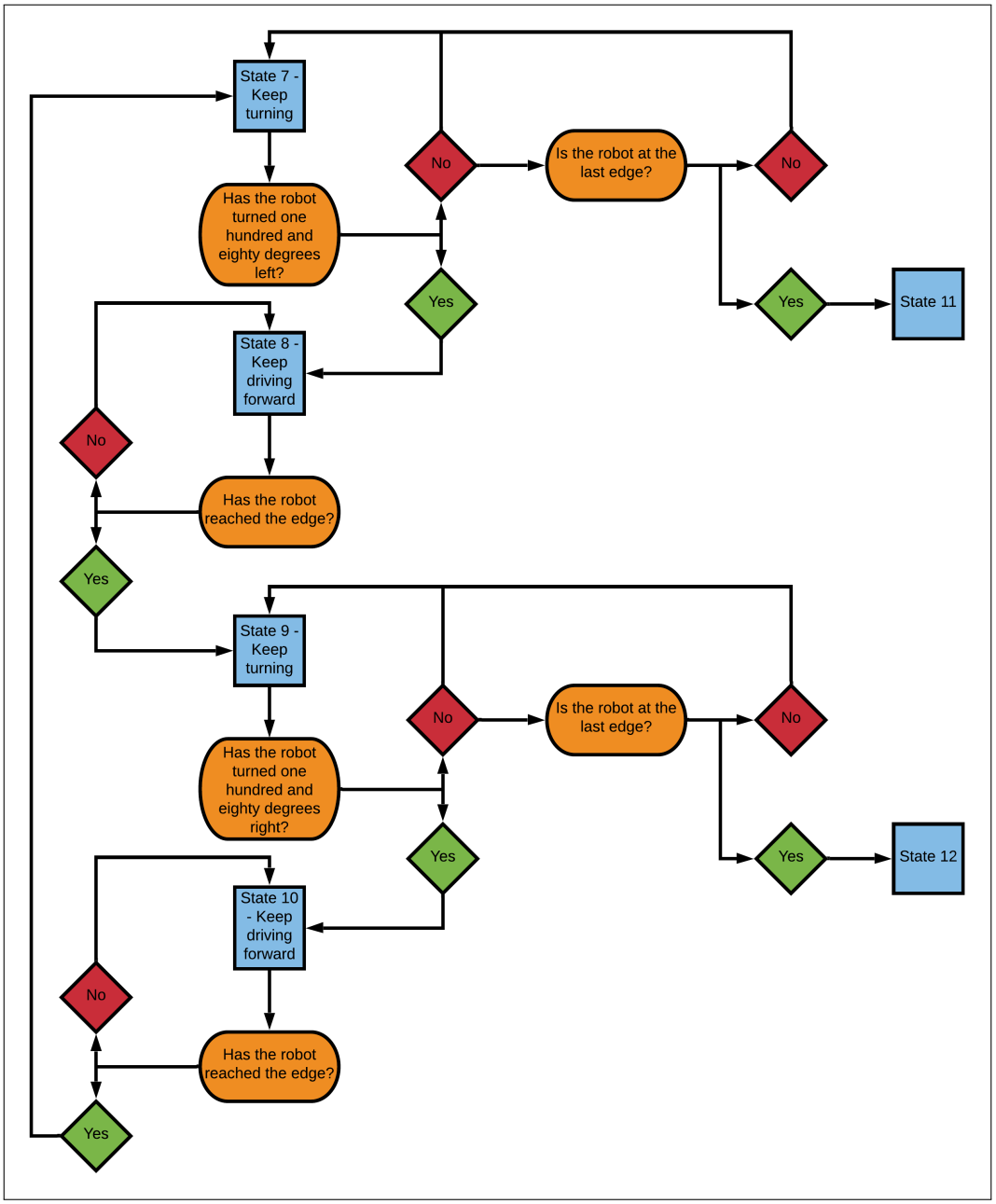


Figure 3.17: Flow chart of states 7-12 for the robot

3.4.7 Heliostat Setup

To aid the robot in finding the edges with the camera, two red lines are fitted to the heliostat. The red edges ensure that the camera can detect and distinguish these edges from the other two, while providing an absolute reference to the robot on the position of a longest edge. The heliostat setup can be viewed in Figure 3.18. These red edges were introduced due to practical challenges in trying to detect the normal edge of the mirror, with false edges being introduced from the surroundings.

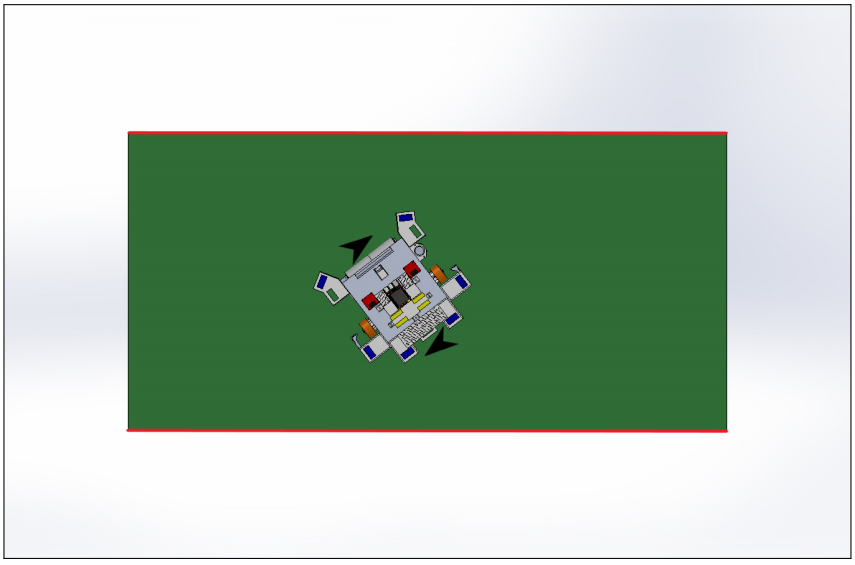


Figure 3.18: Heliostat with red edges. These edges allow the robot to differentiate the heliostat edges from other lines in the environment

3.4.8 ROS Implementation

The robot uses multiple ROS nodes to execute the previously described navigation algorithm. The layout of the ROS nodes can be seen in Figure 3.19. The robot receives real-time data from the camera, ultrasonic sensors, and wheel encoders to navigate the heliostat. The `raspicam_node` takes the image from the camera, and publishes it to the `hsv_color_filter` node, which eliminates all areas in the image that are not red. This image was then pushed to the `hough_line` node, which detects all lines from the red areas of the image. Once these lines are detected, they are published to the `linereader` node, which was where all the data was processed. The `linereader` node receives data from the ultrasonic and encoder publishers, each named to the respective ultrasonic or encoder position. The names in Figure 3.19 which are not circled by bubbles, are the topics on which the data was being published.

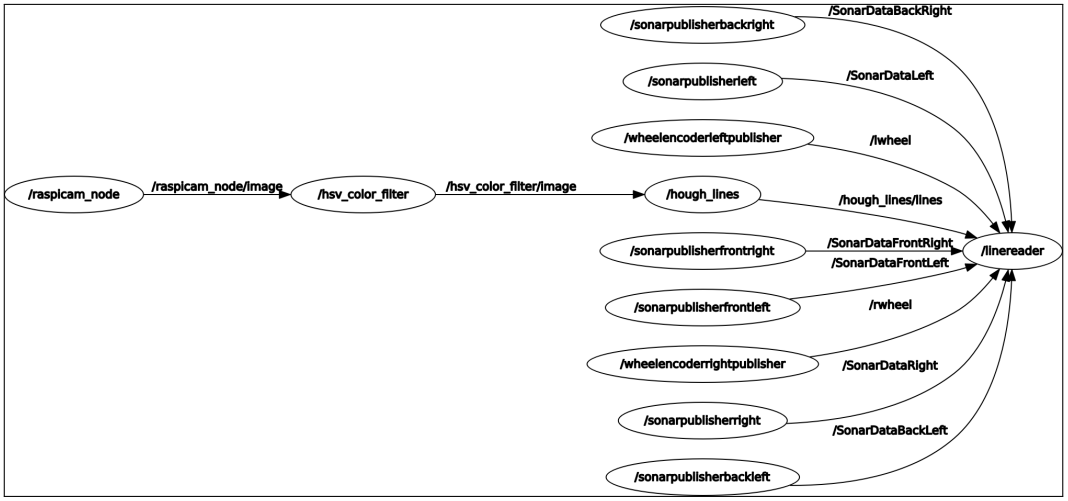


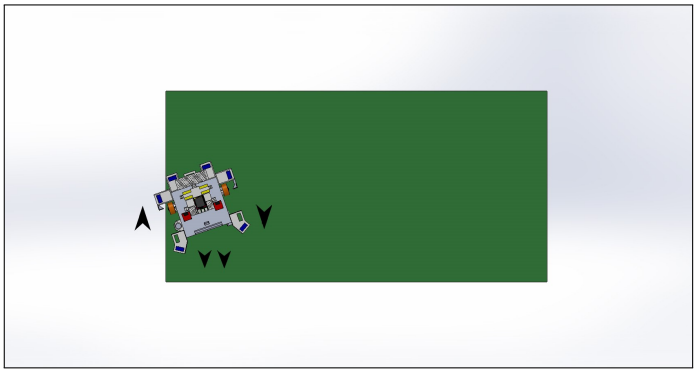
Figure 3.19: ROS node layout, the bubbles represent ROS nodes and the lines are the topics on which they communicate

The `raspicam_node` was set to record at 5 frames per second at each of the test resolutions due to computational limitations. The exposure mode for the camera was set to automatic. The `hsv_color_filter` node was used to ensure that only red lines were seen by the camera. The limits for the node and the (h, s, v) values were defined as $(10 - 350, 128 - 256, 134 - 256)$. The sonar nodes and the encoder nodes were set to sample data at a rate of 5Hz, or once every 200 milliseconds. Lines that were found in the top third of the image frame were disregarded by the `linereader` node, as they were too high to be on the heliostat surface. The `hough_line` node parameters were set to a Hough threshold of 100, a maximum line gap of 50 pixels and the minimum line length as 450 pixels. ρ and θ for the Hough line detection were both left at 1. The queue size for all the ROS data was set to 1, to ensure only the most recent data was being used by the robot.

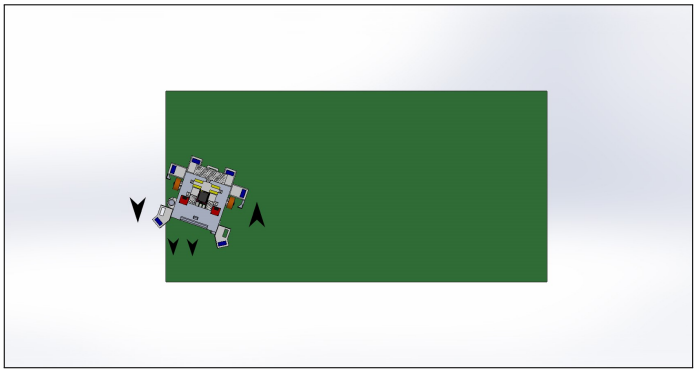
3.4.9 Algorithms

The code used for the robot was extensive, but pseudo code is provided in Appendix B to give insight to the robot navigation. As discussed in 3.4.6, the movement of the robot was broken up into various states. These states are also used to program the robot. The robot can only move from one state to the next once a criterion in each state was met. For navigation, it was important to understand a few of the smaller processes that make up the overall system. The first was the one that allows the robot to drive along the edges. It makes use of the ultrasonic sensors on the side of the robot to detect the edge. The algorithm to drive along an edge on the right side of the robot are given below. The front left ultrasonic was used to stop the motion as it means the robot has reached a corner when both front sonars are over the edge. The edge driving

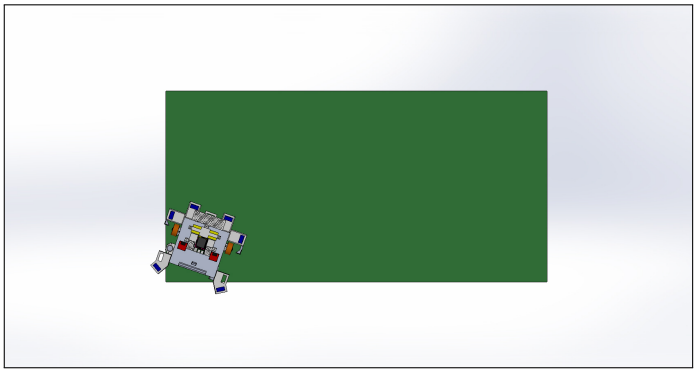
algorithm is given visually in Figure 3.20. Pseudo code for the algorithm is also given in Appendix B in Algorithm 4. The algorithm was also used for the left side of the robot, for any circumstances when it ends in state 12 as in Figure 3.15f



(a) When the front right ultrasonic sensor was over the mirror, the robot drives forward and to the right



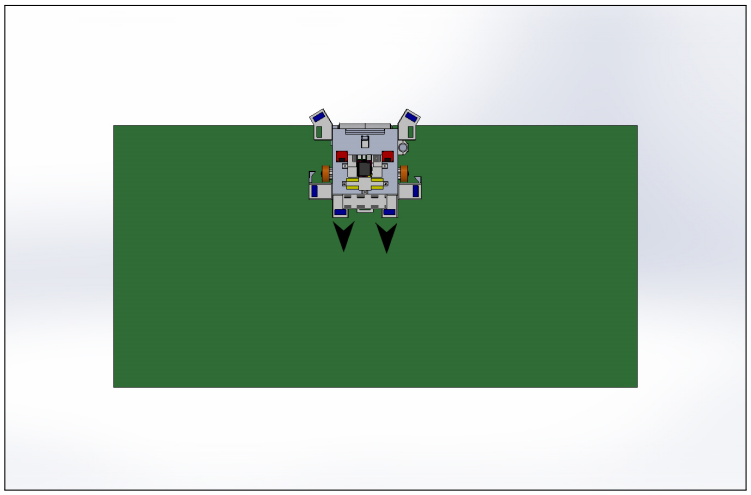
(b) When the front right ultrasonic sensor was off the mirror, the robot drives forward and to the left. If the back right ultrasonic sensor was also off the mirror the robot turns left faster



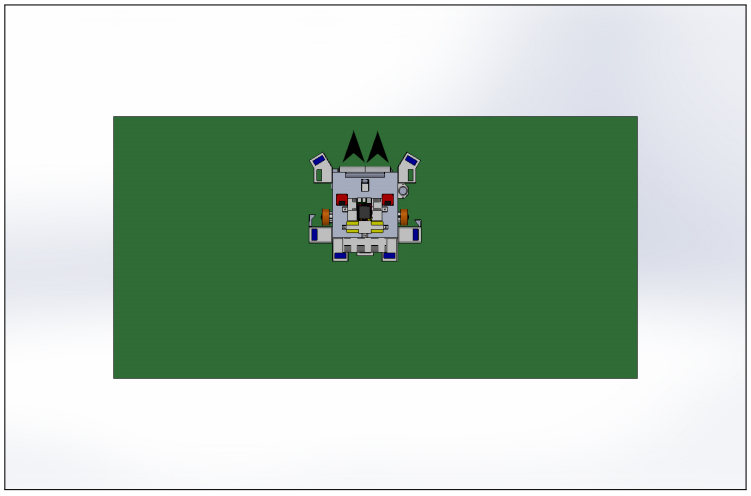
(c) When the robot reaches the corner of the mirror, the edge driving algorithm was finished

Figure 3.20: Visual explanation of the edge driving algorithm

A square up algorithm was developed for both the front and back sets of ultrasonic sensors. This was done to ensure the robot could square up to the edge when camera data was not obtained. The algorithm is explained visually in Figure 3.21, and the pseudo code for the front square up algorithm can be seen in Algorithm 3 in Appendix B.



(a) When both front ultrasonic sensors are off the mirror, the robot reverses



(b) When both front ultrasonic sensors are on the mirror, the robot drives forward

Figure 3.21: Visual explanation of the square up algorithm

When the front right sensor was on the mirror and the front left sensor was off the mirror, the robot will rotate left on the spot. When the front right sensor was off the mirror and the front left sensor was on the mirror the robot will rotate right on the spot. The squaring up algorithm ends when both front sensors either go high or low at the same time. This means they have both gone off the edge or back on the edge at the same moment. The robot was thus square with the edge.

Another important algorithm was that of the gradient finder, which takes all the lines found by the Hough lines, and averages them to provide the gradient of the red edge. The gradient calculator disregards any lines with a Y value less than 300, as these lines are too high to appear on the mirror, and disregards any lines that are too steep, as the red edges should always be horizontal to the robot during the navigation. The gradients are averaged as the Hough lines often detected many lines at the edges. This is discussed in section 4.3.2.

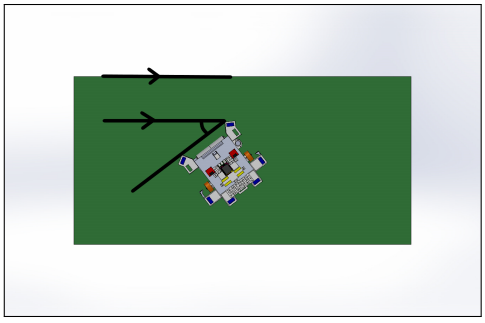
Algorithm 1 Gradient calculator

```
for lines in line message do
    if Y Values ≥ 300 then
        store both X and Y values of the line
    end if
end for
if no lines are found then
    gradient = none
else
    calculate gradient for each line
    if gradient ≤ 0.05 or ≥ -0.05 then
        take the average of these lines
    end if
end if
```

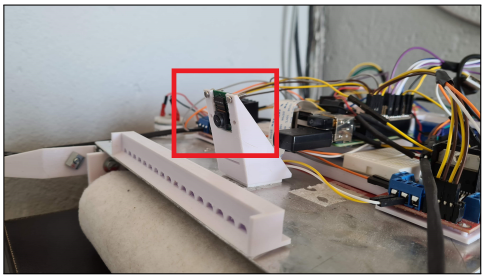
Chapter 4

Camera

The camera, which was chosen as a solution to detect the lines of the heliostat, was fitted to the front of the robot to provide data about the robot's orientation relative to the edge of the mirror. A camera is used as common sensors such as magnetometers cannot be used due to the heliostat frame being metal, and a GPS would not provide the desired accuracy. The ultrasonic sensors and encoders don't provide any absolute reference in the middle of the mirror, so the camera was the only way to obtain data about the robot's position. The accuracy of the camera will determine how well the robot was able to navigate the mirror. The camera was mounted on the front of the robot, inclined at 10° from the vertical. The orientation of the robot as defined as in Figure 4.1a and the position of the camera can be seen in Figure 4.1b. This section aims to determine the accuracy and reliability of using the camera to detect the edges.



(a) Robot orientation defined with respect to heliostat edge



(b) Camera position on robot, the camera was angled at 10° to the vertical

Figure 4.1: Orientation of the robot defined by its position to the edge and the position of the camera

4.1 Pinhole Camera Model

The pinhole camera model was used to relate the 2D image co-ordinates to 3D world co-ordinates. The pinhole camera model can be assumed for any calibrated camera, but to use this method, the extrinsic and intrinsic camera parameters must be known. The extrinsic parameters were determined both experimentally and theoretically to determine which yielded better solutions.

4.1.1 Theoretical Pose Estimation

To determine the robot's orientation to any of the mirror edges, a relationship between the image plane and world plane must be established. The pinhole camera model can be used on any rectified image to determine this relationship using the intrinsic and extrinsic parameters of the camera. The extrinsic parameters are the rotation and translation (pose) of the camera relative to the world, and the intrinsic camera parameters are determined by the physical properties of the camera. The world co-ordinate system was chosen to be directly below the camera, on the surface of the mirror, with the Z axis facing off the mirror, Y axis facing forward and the X axis facing to the right side of the robot, or into the page. The x axis for the camera was chosen to be in the same direction as the world axis, with the y axis aiming down and the z axis out of the camera. The selected axis systems can be seen in Figure 4.2.

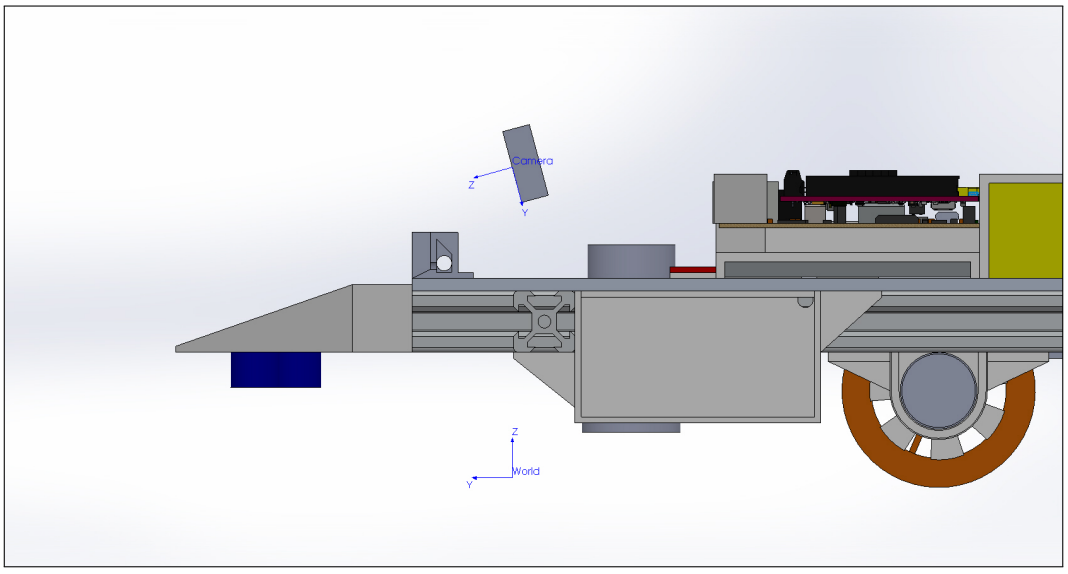
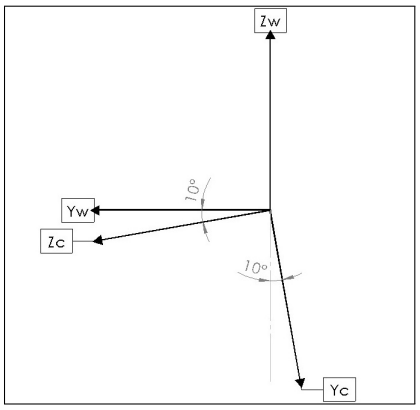


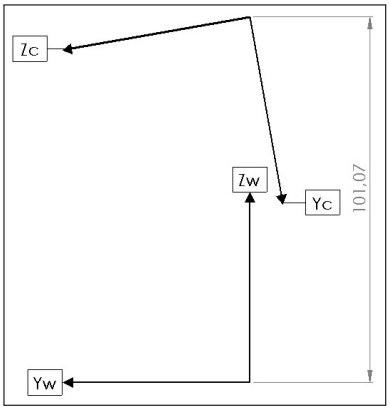
Figure 4.2: Position of the camera co-ordinate system relative to the world co-ordinate system

The rotation and translation matrices of the camera can now be obtained. The rotation matrix was obtained by putting both co-ordinate systems on the

same point, and then computing the camera co-ordinate system in terms of the world co-ordinates. The rotation of the two co-ordinate systems can be observed in Figure 4.3a. The translation matrix was obtained by using the theoretical distance between the two co-ordinate systems, as in Figure 4.3b.



(a) World co-ordinate system, (X_w, Y_w, Z_w), and camera co-ordinate system,(X_c, Y_c, Z_c), at the same origin, this was done to determine the rotation matrix between them



(b) World co-ordinate system, (X_w, Y_w, Z_w), and camera co-ordinate system,(X_c, Y_c, Z_c), with translation included, this was done to determine the translation matrix between them

Figure 4.3: Rotation and translation between camera co-ordinate system and world co-ordinate system

The resulting theoretical rotation and translation matrices for the camera to the world co-ordinate system are given in Equations 4.1 and 4.2 respectively. This was the theoretical pose of the camera.

$$Theoretical\ Rotation = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.174 & -0.985 \\ 0 & 0.985 & -0.174 \end{bmatrix} \tag{4.1}$$

$$Theoretical\ Translation = \begin{bmatrix} 0 \\ 99.53 \\ 17.55 \end{bmatrix} \text{mm} \tag{4.2}$$

The camera would be tested at 3 different resolutions, to determine if resolution was directly related to the accuracy of the robot's orientation. This meant that three intrinsic matrices had to be obtained. OpenCV camera calibration was performed at 1024x576, 1280x960 and 1600x1200. The resulting calibration matrices and distortion coefficients, (rounded to two decimals) for each resolution can be seen in Equations C.1 through C.6 in Appendix C.2.

All the data to use the pinhole camera model in Equation 2.6 has been obtained, allowing the image co-ordinates to be mapped to world co-ordinates

to predict the orientation of the robot. It was also possible to obtain this data using experimental methods. By taking an image of the plane of interest (mirror surface) and knowing the world co-ordinates of points in that image, the opencv function `solvepnp` attempts to find the pose of the camera relative to the world. `Solvepnp` also makes use of Equation 2.6, and thus requires the intrinsic parameters and distortion coefficients (for images which have not been rectified) of the camera. The co-ordinates of points in the world, and the image co-ordinates of these points are also required in predicting the camera pose. The results from the theoretical camera pose and the experimental camera pose will be compared to find the most accurate means of predicting the robot's orientation.

4.1.2 Experimental Pose Estimation

To determine the pose of the camera using `solvepnp`, real world co-ordinates need to be obtained for points in the image frame. To do this, the robot was placed on a flat surface like a heliostat mirror, and points were marked out on the surface. The same co-ordinate system locations seen in Figure 4.2 are used, meaning that the X and Y values of the real world points were measured from the origin of the world axis fixed below the robot. All the Z values were assumed to be 0, as the measurement was done on the $X - Y$ plane. The experimental setup to obtain the pose of the robot can be seen in Figure 4.4. The world co-ordinate system in Figure 4.4 was shifted forward for display purposes, in the experimental data the axis was positioned below the camera.

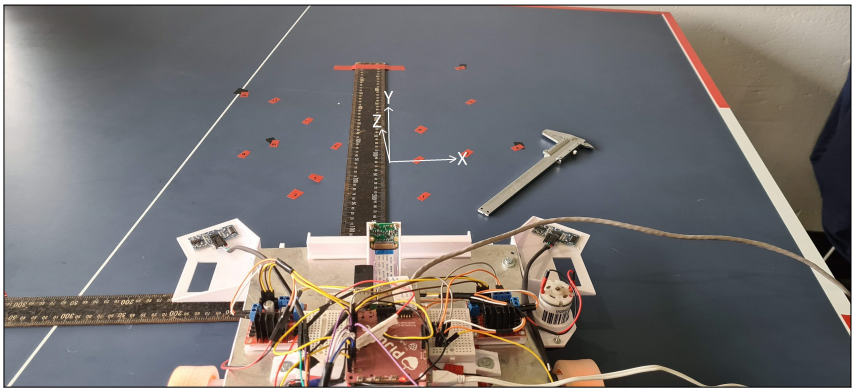


Figure 4.4: Experimental setup for obtaining the pose with solvepnp

For each resolution, eight points are used to obtain the experimental pose of the camera. `Solvepnp` requires only four points, but more points were used to eliminate potential error. The world co-ordinates, (X, Y), and their respective image co-ordinates, (u, v), for each resolution can be seen in Appendix C Table C.1. The u co-ordinate was measured from the left side of the image

while the v co-ordinate was measured from the top of the image as in Figure 4.5. The (X) values are measured from the line of centre of the robot, negative values are points to the left and positive values are to the right. The (Y) values are measured from the front of the robot.

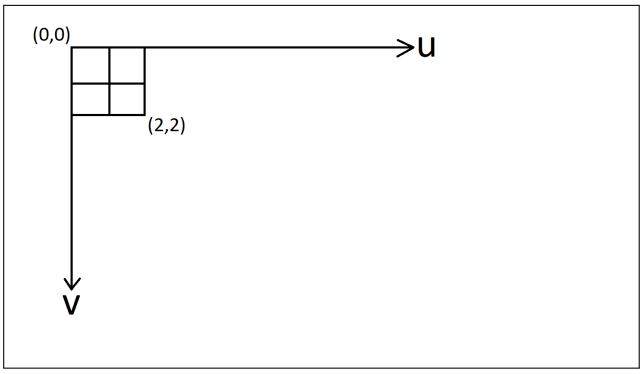
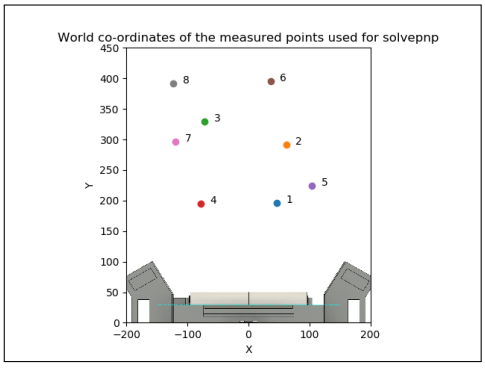
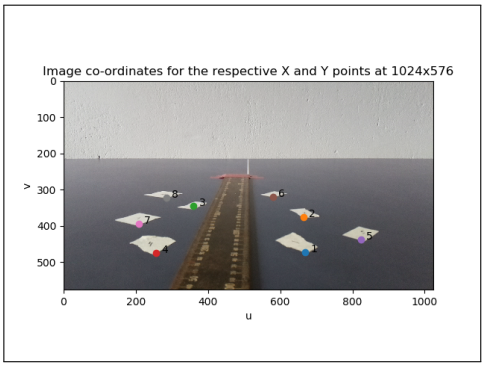


Figure 4.5: Image co-ordinates, (u, v), the origin lies at the top left of the image, u increases as one moves to the right of the image and v increases when moving down the image frame

The world co-ordinates and image co-ordinates for 1024x576 are show in Figure 4.6 to further explain the concept and significance of Table C.1.



(a) The world points, (X, Y), are measured from below the camera, on the mirror plane. The eight points used for solvepnp were spaced in front of the robot in the camera's field of view



(b) The image co-ordinates, (u, v), of the world points, (X, Y), from 4.6a. This was the camera perspective on the points used for solvepnp

Figure 4.6: To make use of solvepnp, eight world points on the mirror had to be mapped to eight image points for each resolution. These images provide the perspective for the 1024x576 resolution, the process was repeated for all three resolutions and the data are shown in Table C.1

Using `solvepnp`, and the data from Table C.1, the rotation and translation matrices for the experimental data at each resolution was obtained. The equations are given in Appendix C.2. Comparing Equations C.7 through C.12 to Equations 4.1 and 4.2, it can be seen that the theoretical rotation and translation matrices have similar values to the experimentally determined values at each resolution.

The resulting experimental rotation and translation matrices for the camera to the world co-ordinate system at 1024x576 are given in Equations 4.3 and 4.4 respectively. This was the experimental pose of the camera at 1024x576.

$$Experimental\ Rotation = \begin{bmatrix} 0.9998 & -0.0193 & -0.0101 \\ -0.0135 & -0.1838 & -0.9829 \\ 0.0171 & 0.9828 & -0.1841 \end{bmatrix} \tag{4.3}$$

$$Experimental\ Translation = \begin{bmatrix} 2.3454 \\ 92.6441 \\ 54.8349 \end{bmatrix} \text{mm} \tag{4.4}$$

4.1.3 Camera Matrices

The data from the theoretical and experimental pose estimation (extrinsic camera parameters) are accumulated to obtain the pinhole camera models for all three camera resolutions. The intrinsic matrices obtained from opencv camera calibration are used for both the theoretical and experimental models, and the theoretical rotation and translation matrices are be used for all three resolutions. The resulting theoretical matrices for each resolution, (rounded to two decimals) can be found in Appendix C.2, Equations C.13 to C.15. Similarly, the resulting experimental matrices (rounded to two decimals) can be found in Appendix C.2, Equations C.16 to C.18.

To make it clear, the experimental camera model means that the extrinsic parameters of the camera were determined experimentally using solvepnp, and the theoretical camera model means that the extrinsic parameters were determined theoretically using analytical rotation and translation matrices. The intrinsic parameters for all camera models were found with camera calibration.

4.2 Expected Accuracy

The matrices obtained in the previous section shown in Appendix C provide a relationship between the 2D image plane and the 3D world plane. It was important to know how accurate these relationships can be for each camera resolution, as it will affect the overall accuracy of the robot when determining its orientation relative to the heliostat edges. To get a sense of the accuracy, the image co-ordinates from Table C.1 were fed back into the pinhole camera Equations C.13 to C.18 where the predicted real world co-ordinates for both the experimental and theoretical pose models are returned. These predicted co-ordinates are compared to the measured co-ordinates, and the percentage errors are analysed in Appendix C in tables C.2 to C.4.

Table C.2 shows that the theoretical predictions of the camera have a higher average error than the experimental predictions when predicting the position of known world co-ordinates from their image co-ordinates. The average Y errors for both models are higher than that of the average X errors, this was likely due to the fact that there are more horizontal pixels than vertical pixels at this resolution. The maximum error for the theoretical model was also substantially higher than that of the experimental model for both the X and Y values.

The points in Table C.2 are plotted on the $X-Y$ axis in Figure 4.7. The Z value for all the points was 0, as all the points lie on the floor plane in front of the robot as seen in Figure 4.4. The plot shows that the experimental model provides a more accurate prediction of the actual values of the points, although they are not completely accurate. This plot suggests that both models will have some degree of inaccuracy when trying to predict the orientation of the edges of the heliostat, but it was likely that the theoretical model will be most inaccurate.

Table C.3 in Appendix C shows the same result as C.2, the errors for the theoretical model are greater for both the X and Y predictions, although the average errors were reduced with the higher resolution. The average errors for the experimental model did not change significantly.

Table C.4, also in Appendix C reinforces the fact that the experimental predictions are more accurate than the theoretical predictions. The average error for the theoretical model increased from 1280x960, suggesting that the theoretical models are not a good reflection of the actual pose of the camera. The experimental prediction errors are still smaller than that of the theoretical models and seem to grow smaller with higher resolution.

The data from the tables above show that the errors found using the experimental poses of the camera are significantly lower than the errors found using the theoretical pose. It was evident that the Y errors are larger than the X errors for both the theoretical and experimental data, likely due to the fact that the horizontal resolutions are higher than the vertical resolution for each set of data. It was likely that estimating the pose of the robot will be

more sensitive to changes in the vertical co-ordinates rather than changes in the horizontal co-ordinates. Due to this, it was also likely that the accuracy of predicting the robot orientation will increase with resolution, as more vertical pixels will allow for more accurate prediction of the world co-ordinates.

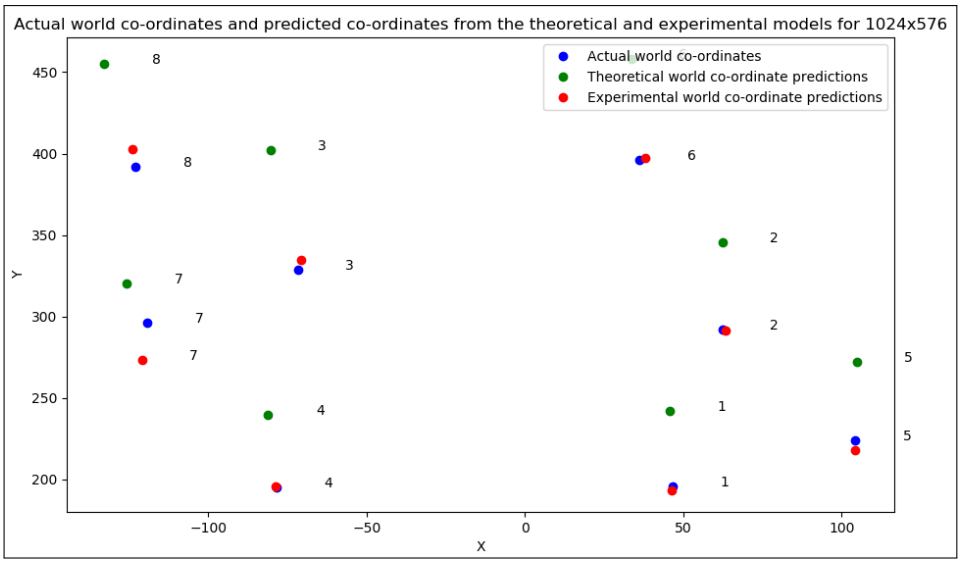
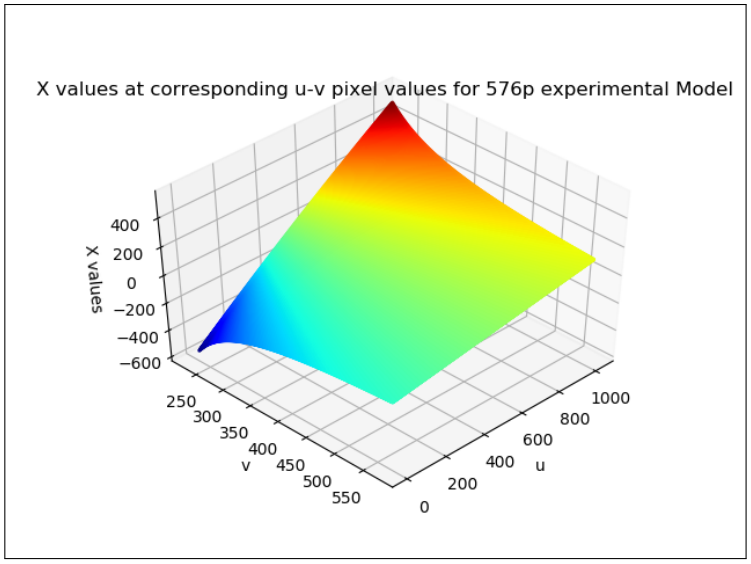


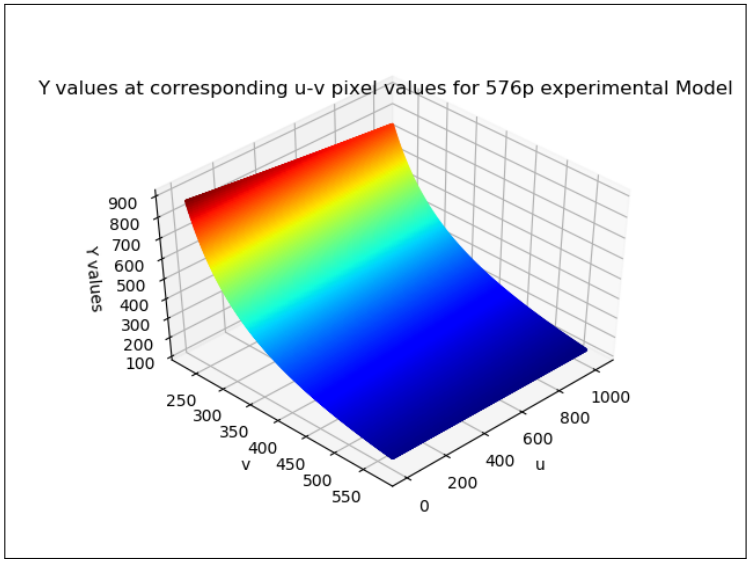
Figure 4.7: The actual world co-ordinates are compared to the co-ordinates predicted by the theoretical and experimental models. The known world co-ordinates have a respective pixel co-ordinate which are given to the experimental and theoretical camera matrix to return the predicted world co-ordinate. It can be seen that finding the extrinsic parameters of the camera using solvepnp (experimentally) yields better results than using the analytical camera pose (theoretically) as they more accurately represent the actual world co-ordinates

Using the matrices in Equations C.13 to C.18 (Appendix C.2, the X and Y values for each image pixel u, v can be plotted on a 3D surface when $Z = 0$. The v co-ordinates were limited to the bottom 60 percent of the image frame, as this was the area where edges are detected. Figure 4.8 shows this relationship for the experimental model at 1024x576. In Figure 4.8a, one can see that points closer to the robot, (higher v value) have a low gradient. When moving up the image frame, (lower v value) the gradient of X values begins to increase. The gradient of the X values also become steeper on the sides of the image (high and low u values). This gradient suggests that lines found in the middle of the image, closer to the robot will likely be more accurate than lines found further away. In Figure 4.8b, it can be seen that the Y values grow exponentially for points higher in the image, (low v values). This suggests that the further a line was detected, the more sensitive the orientation of the line will be. These plots are similar for all of the Equations C.13 to C.18, and the

results for the rest of the resolutions, both experimental and theoretical can be seen in Appendix C.3 figures C.1 to C.3.



(a) X values at corresponding pixel co-ordinates for the 1024x576 experimental model. Locations closer to the robot have a lower gradient for the corresponding X values. Points higher in the image frame have a higher gradient in their X values, suggesting larger errors the further away from the robot a line lies



(b) Y values at corresponding pixel co-ordinates for the 1024x576 experimental model. Locations closer to the robot have a lower gradient for the corresponding Y values. Points higher in the frame have a higher gradient, and grow exponentially. Large errors are extremely likely at higher points of the camera

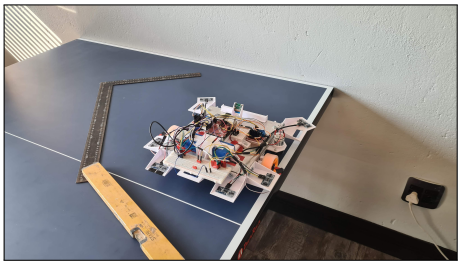
Figure 4.8: $X - Y$ plots for $u - v$ values for 1024x576 experimental model. These plots show how world X and Y co-ordinates are affected by the pixel co-ordinates It was clear that there was a non-linear relationship for both co-ordinates

4.3 Camera Accuracy

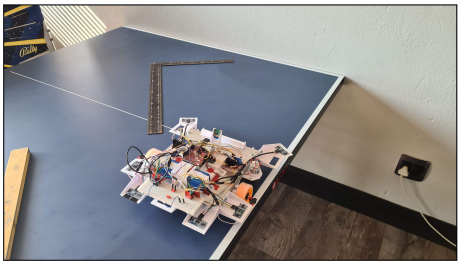
This section discusses the experimental methods followed to determine the accuracy of the camera in detecting the edges of a heliostat. The results are also provided which discuss this accuracy. Camera accuracy was critical to the system as it has a direct impact on the capability of the robot to navigate the heliostat when it was in the middle of the mirror.

4.3.1 Experimental Setup

To analyse the capability of the robot in determining its orientation relative to the edge, the robot was placed stationary on a flat surface with the camera facing the edge at three different distances (300 mm,600 mm,1200 mm) from the line. It was rotated from 0° to 20° at 2° increments at each distance. Each time the robot was rotated, an image was taken of the edge in the frame at each of the three resolutions. To place the robot at specific orientations to the line, a large protractor was used to ensure that the robot was at the correct orientation. The experimental setup can be observed in Figure 4.9.

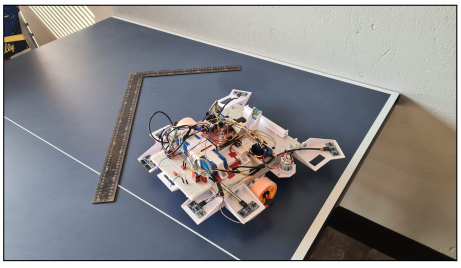


(a) Robot at closest distance from the test edge positioned at 0°



(b) Robot at middle distance from the test edge positioned at 0°

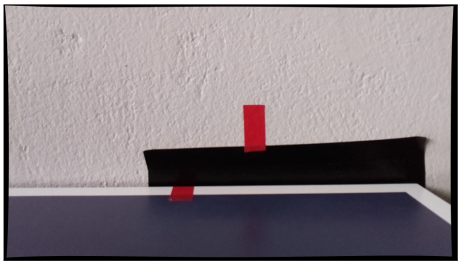
(c) Robot at furthest distance from the test edge positioned at 0°



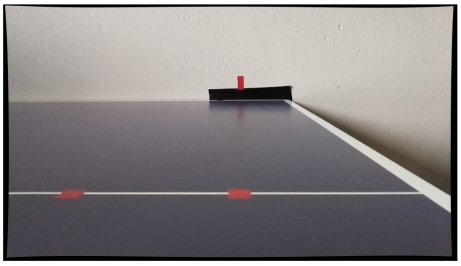
(d) Robot at closest distance from the test edge positioned at 10°

Figure 4.9: Experimental setup of the robot to determine the accuracy of the Hough line detection. The robot was placed at three different distances from the test edge, at angles ranging from 0° to 20° at 2° increments to see the effects distance may have on the results

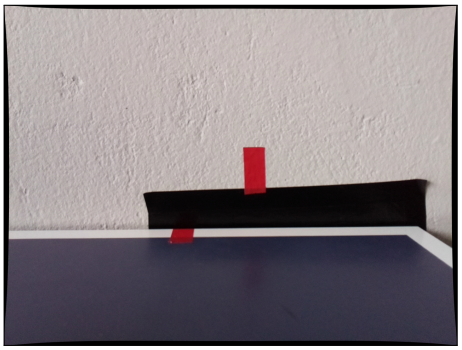
The resulting images were then rectified using the respective camera parameters and distortion coefficients for each resolution, to ensure distortion would have no effect on the results. Examples of the images taken can be seen in Figure 4.10.



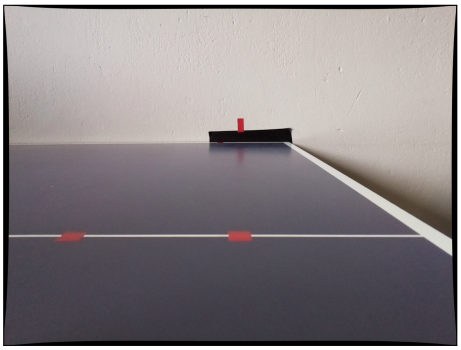
(a) Image taken at 1024x576 from the closest point at 0°



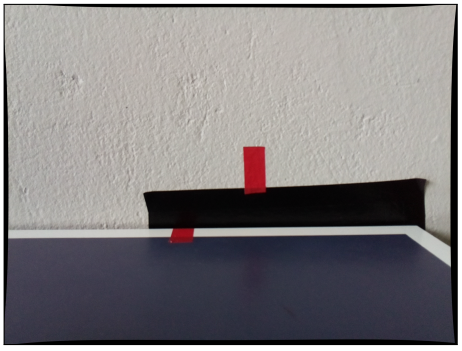
(b) Image taken at 1024x576 from the furthest point at 0°



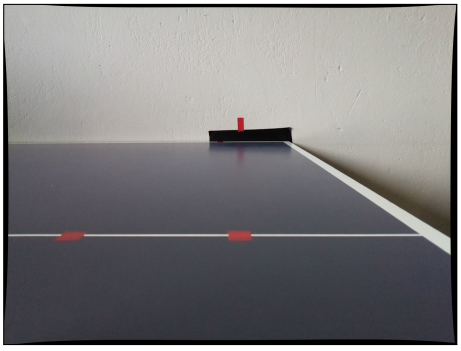
(c) Image taken at 1280x960 from the closest point at 0°



(d) Image taken at 1280x960 from the furthest point at 0°



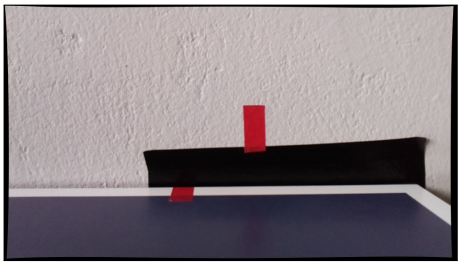
(e) Image taken at 1600x1200 from the closest point at 0°



(f) Image taken at 1600x1200 from the furthest point at 0°

Figure 4.10: Resulting images taken by the robot when positioned at 0°, at the closest and furthest point for all three resolutions

Once all the images had been obtained, the same section of the line was used in predicting the orientation. Each image was made to focus on the area of the line in-front of the dark area to ensure the Hough line detection would be consistent in analysing the same area across all the images. Instead of cropping the images, a dark area was placed over the image around the line. This crop ensured that the image resolution did not change from cropping when analysing the lines. An example of one of the processed images are given in Figure 4.11.



(a) Image before being focused on the line, which represents a heliostat edge

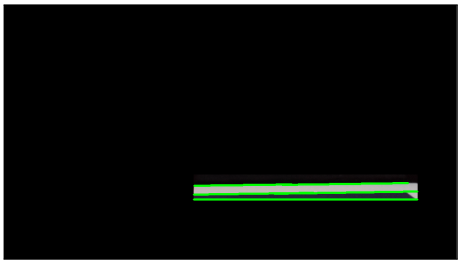
(b) Image after being focused on the line, so that the Hough line detection may disregard the surroundings

Figure 4.11: Post-process of images to ensure line detection

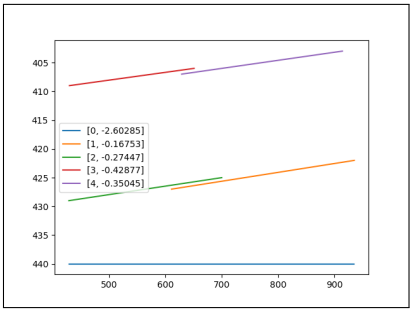
4.3.2 Processing Method

To obtain the angles taken for each image, a python script was developed which extracted all the lines found in the image using Hough Lines. These lines were then converted to real world co-ordinates using Equations C.13 to C.18, and the angles of each line relative to the robot were calculated using these co-ordinates. Multiple lines were often detected in the images, so the line that best fit the known real world angle was chosen, as this would show the capability of the system in obtaining its angle relative to the edge if the correct line are found. An example of a processed image, as well as the resulting lines can be seen below in Figure 4.12.

Once each image was processed, the angles found were compared to the real angle and the closest value was stored. The Hough threshold was adjusted to see what effect it may have on the results. Thresholds of 50, 100 and 150 were used. The other Hough parameters (minimum length, maximum line gap, rho and theta) were all fixed at the different thresholds. These values were not adjusted as it would introduce too many variables. The values were fixed as follows, rho = 1, theta = 1, minimum length = 100, maximum line gap = 100. The horizontal lines detected from the black border were not considered.



(a) Image of the edge processed with Hough lines



(b) Resulting real world lines and angles obtained from the Hough line detection

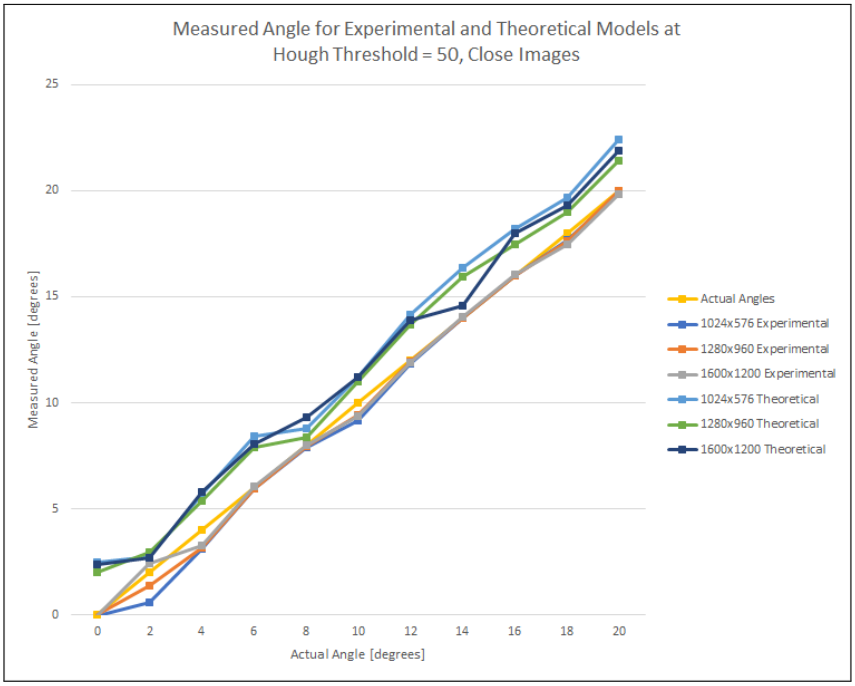
Figure 4.12: Post Processing Example of the Images Obtained from Orientation Testing

4.3.3 Results - Experimental vs Theoretical Models

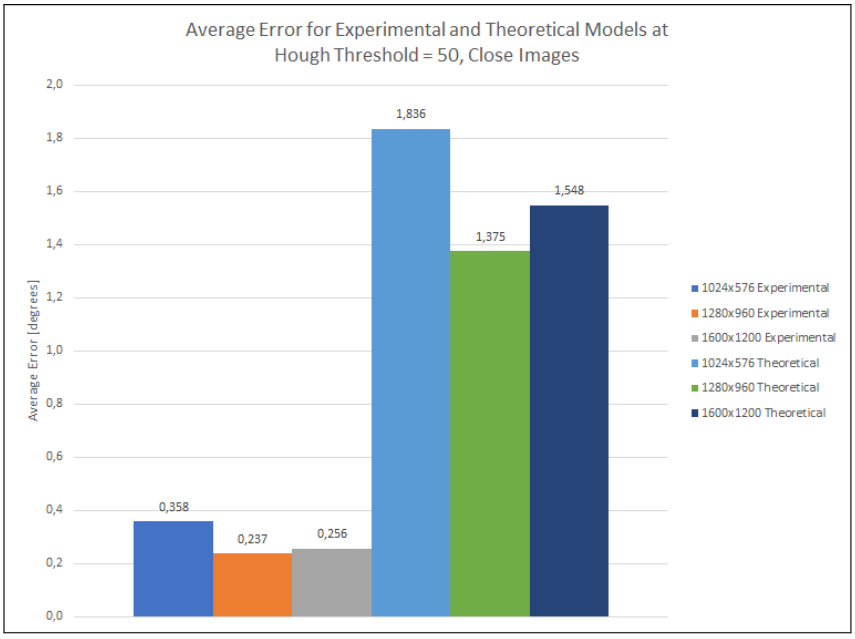
To determine the capability of using the camera to obtain the orientation of the robot, the effects of distance from the edge, image resolution and Hough threshold are considered. Using the processing method in section 4.3.2, each image was evaluated by Hough line detection. The results are plotted and discussed for various conditions, and the absolute errors for these conditions are also evaluated. The experimental results are compared to the theoretical results, to see which model more accurately fits the data, this model was then used to analyse the rest of the data.

Figure 4.13a plots the measured angles against the actual angles for the theoretical and experimental models at all three resolutions at the closest distance to the line. The plot shows that the theoretical model was more inaccurate than the experimental model when predicting the orientation of the lines. Figure 4.13b plots the average error for Figure 4.13a, and shows that for the various resolutions, the theoretical data has significantly higher average errors than the experimental data. This was verified with Figure 4.13c which shows that the average errors for the theoretical model at 1024x576 are substantially higher at all three distances than the experimental model for the same resolution.

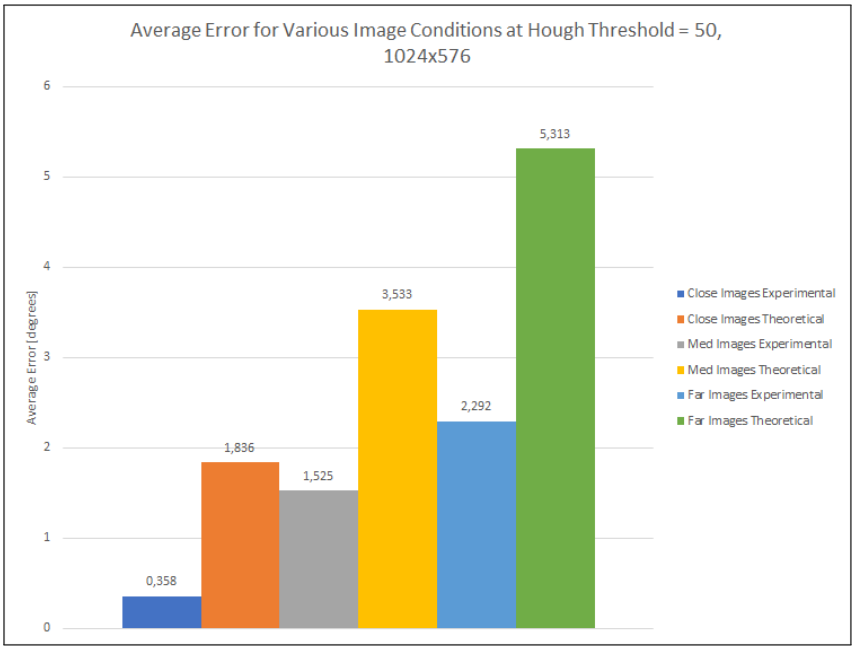
The theoretical model provides higher errors at all resolutions, thresholds, and distances. The theoretical model will no longer be analysed for the data, as the experimental model was more accurate with all scenarios.



(a) Measured angles for experimental and theoretical models at Hough threshold 50 for close images. The theoretical model was significantly more inaccurate than the experimental model for all resolutions



(b) Average error for experimental and theoretical models at Hough threshold 50 for close images. The theoretical model has significantly higher errors than the experimental model, regardless of the camera resolution

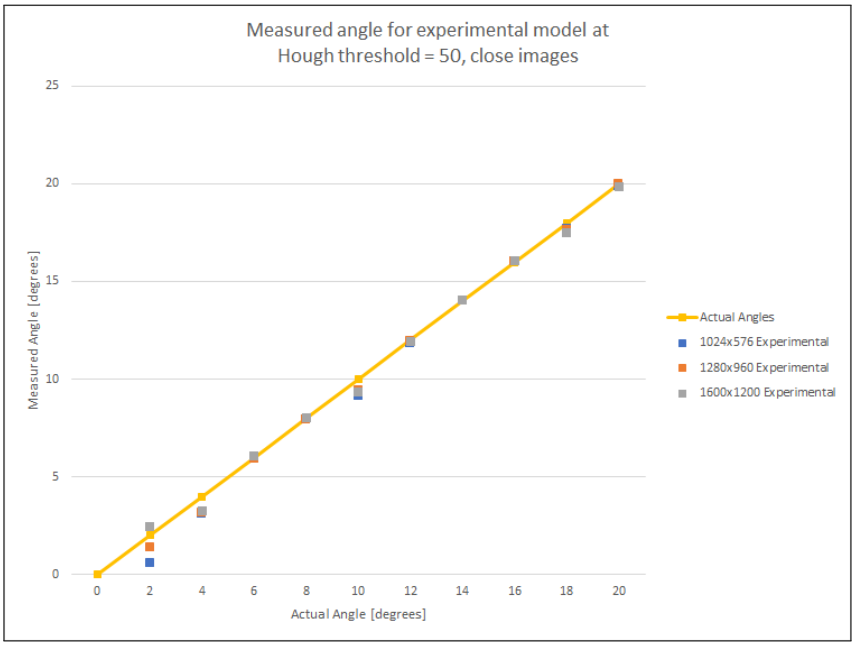


(c) Average error for experimental and theoretical models at Hough threshold 50 for 1024x576. The theoretical model has significantly higher errors than the experimental model, regardless of how far the lines are from the camera

Figure 4.13: Graphs of angle vs distance at Hough threshold 50. The theoretical model under performs in all scenarios, meaning that only the experimental model will be used to analyse the data from Hough line detection

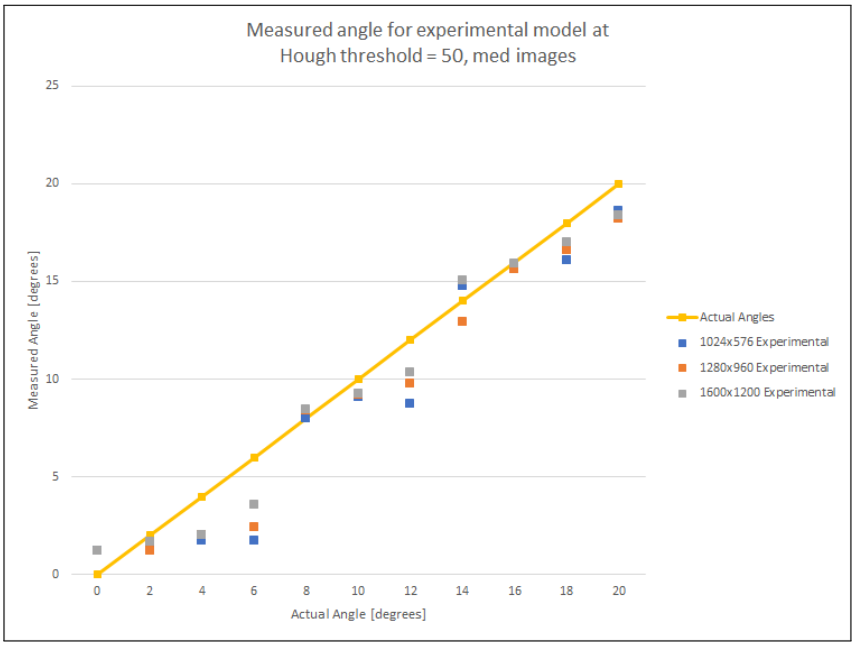
4.3.4 Results - The Effect of Image Resolution

The first set of graphs in Figure 4.14 plots the obtained angles for all the resolutions at various distances with the Hough threshold fixed at 50. It was immediately evident that when the robot was close to the line, the resolution does not have a large effect on the accuracy of the line detection as seen in Figure 4.14a. Looking at figures 4.14b and 4.14c, it can be seen that higher resolutions do provide a more accurate estimate of the angle for most of the data. Additional data to support a higher resolution providing a better estimate of the angles can be seen in Appendix C.3, from figures C.4a and C.4d.

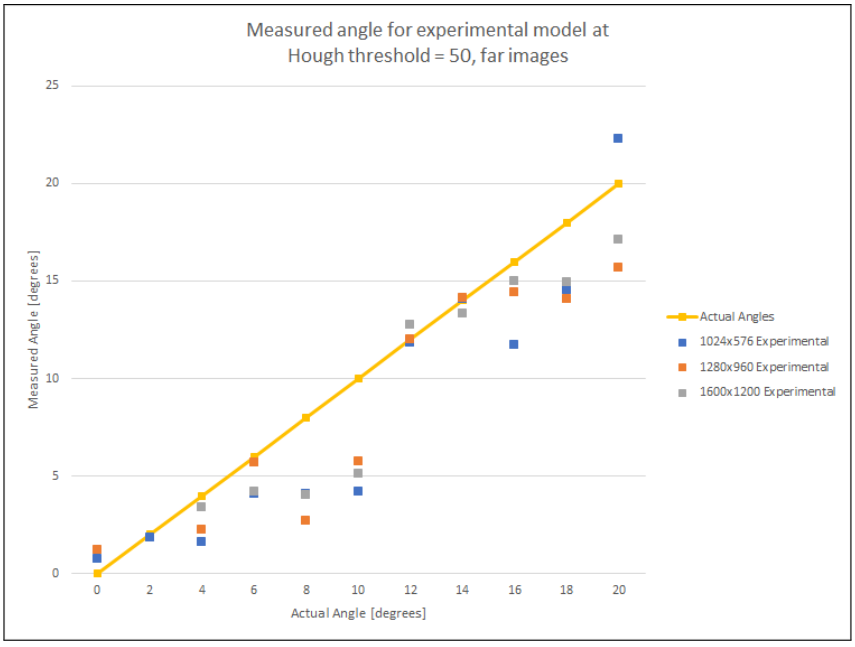


(a) Measured angles for experimental model at Hough threshold 50 for close images. When the robot was positioned close to a line, the effects of resolution are small. All three resolutions provide a good measure of the true orientation of the edge

Figure 4.15 provides the average error for the graphs in Figure 4.14. The plot suggests that a higher resolution ensures a lower average error for the estimation of the line, with the exception of a lower error in the middle resolution for the close image. The graph suggests that a higher resolution will ensure a lower average error, however at the furthest distance it can be seen that the average error for all the resolutions are similar. Figure C.5 in Appendix C.3 show that for all Hough thresholds a higher resolution creates less average error for each distance from the line. The average error was highest when the camera was the furthest away from the line, and the lowest when closest to the line for all the data. This was what was expected from discussing figures 4.8 previously.



(b) Measured angles for experimental model at Hough threshold 50 for medium images. At a further distance, the effects of a higher resolution are apparent. A higher resolution provides more accuracy when the robot was further from the line



(c) Measured angles for experimental model at Hough threshold 50 for far images. A higher resolution leads to a more accurate prediction of the orientation of a line, but at the furthest point the accuracy of all three resolutions decreases significantly

Figure 4.14: Graphs of angle vs distance at Hough threshold 50, analysing the effects of resolution

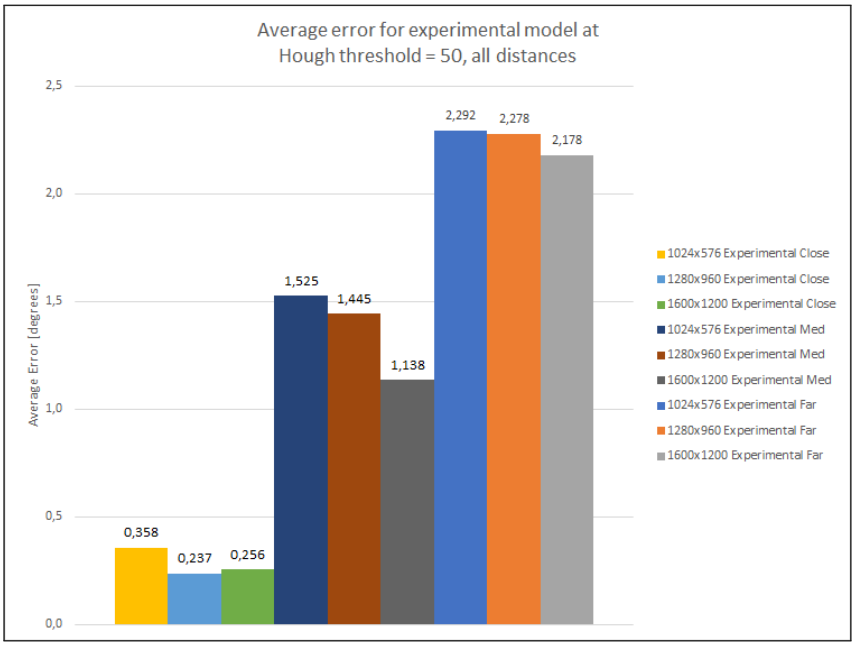
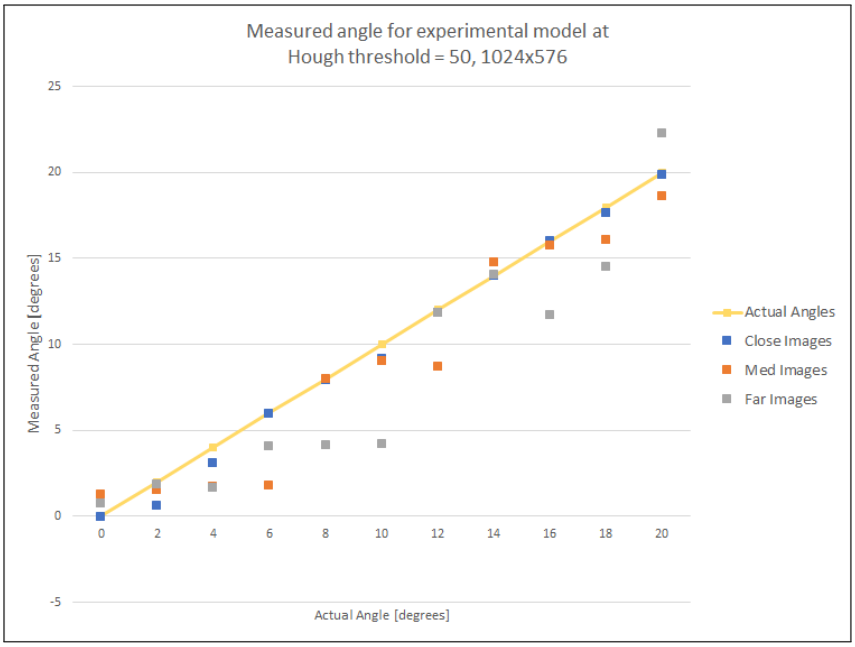


Figure 4.15: The average error for various resolutions at Hough threshold 50 was plotted. The higher resolutions for the close distance images provide a more accurate estimation of the orientation of the line, although the middle resolution appears to be most accurate. This error was likely due to an outlier in the data. The average error for the medium distance images suggests a higher resolution provides a better estimation of the real orientation of the lines when the robot was further away from the edge. The average error for the far images suggests the effects of resolution at the furthest distance seem to be less prominent. There was no major difference between the average errors for the three resolutions. This suggest that there may marginal gains at higher resolutions when further away from the line

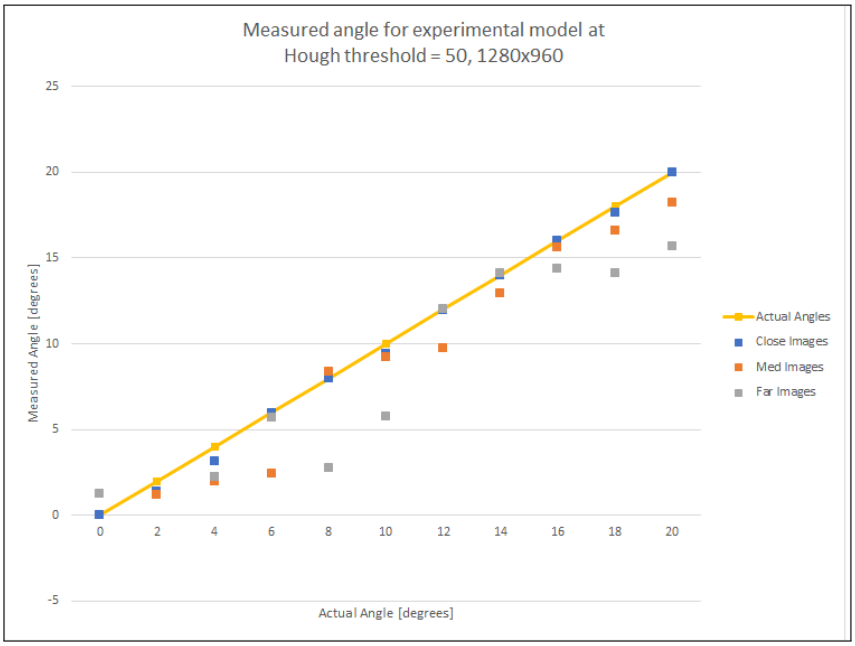
4.3.5 Results - The Effect of Distance from the Edge

Figure 4.16 plots the measured angles for all three resolutions at various distances from the line. For each resolution, the data when the camera was closest to the line was the most accurate. The accuracy of the far images was unpredictable for all resolutions, suggesting that the line detection will struggle to obtain the robot's orientation when the robot was far away from an edge. The average values for the errors are also compared, and data for Hough thresholds of 100 and 150 are found in the Appendix C.3 Figure C.6, which show the same trend. It was also noteworthy to mention that higher Hough line thresholds return no lines for some of the further images. Figures 4.16b and 4.16c do not have all data points at the lower angles.

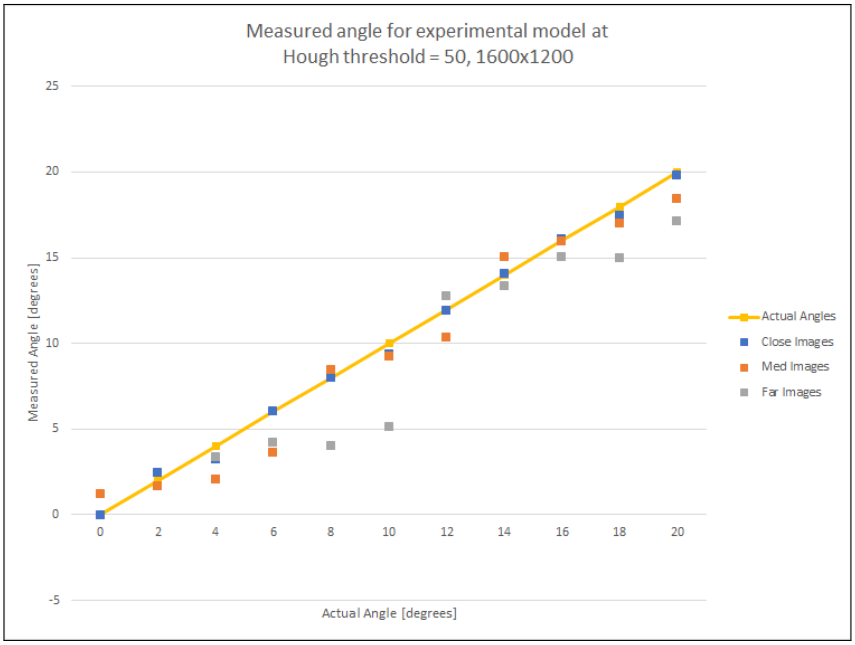
Figure 4.17 plots the average errors for Figure 4.16. The average error of the angles increases with distance to the line, for all resolutions and regardless of the Hough threshold, (1280x960 and 1600x1200 can be found in Appendix C.3 Figure C.7). The far images return the same average error for all three resolutions; however, the low resolution does have significantly higher errors at the close and far positions. The medium and high resolutions return similar average errors at the close and medium distances.



(a) Measured angles for experimental and theoretical models at Hough threshold 50 for 1024x576. The further away from the line the robot was, the less capable it was of predicting the orientation of the line



(b) Measured angles for experimental and theoretical models at Hough threshold 50 for 1280x960. This result was consistent with that of 1024x576, the further the robot was from the line the higher the prediction errors



(c) Measured angles for experimental and theoretical models at Hough threshold 50 for 1600x1200. Although this resolution was the most accurate, a higher distance still causes a higher error in the prediction of the line orientation

Figure 4.16: Graphs of angle vs distance at threshold 50, analysing the effects of distance

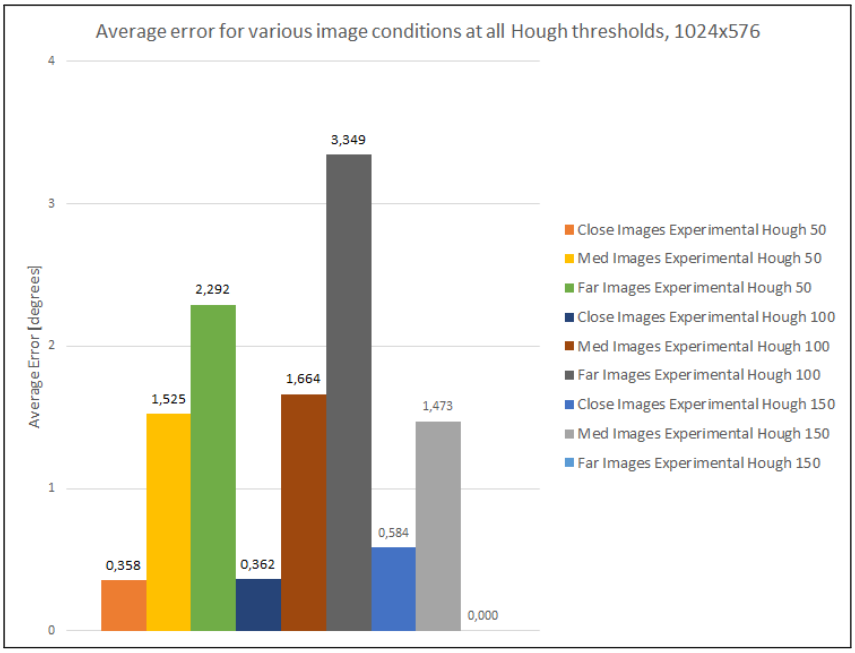


Figure 4.17: The average error for various image conditions at all Hough thresholds was plotted at 1024x576. The average error for line prediction distance increases significantly with distance, regardless of the Hough threshold. The lines closest to the camera have the best result. The error at the far distance was similar regardless of resolution, but no lines were found at the far distance with a Hough threshold of 150

4.3.6 Results - The Effect of Hough Threshold

Figure 4.18 contains plots for the various Hough thresholds at each resolution for the close images. The plots show that an increase in the Hough threshold causes an increase in the average error of the estimated lines. It was also worth noting here that an increase in the resolution leaves a lower error at the same Hough thresholds. The higher Hough thresholds, however, detect a smaller number of lines, which are expected because the requirements to form a line require more bin votes for a line to be declared. The number of lines found for each Hough threshold are plotted on the following page.

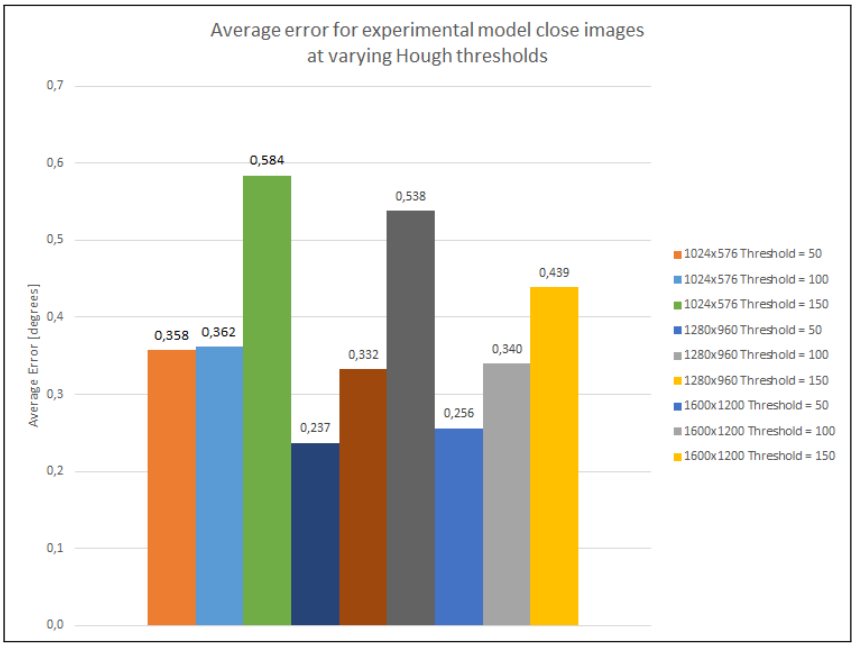


Figure 4.18: The graph plots the average error for various Hough thresholds at all three resolutions for the close images. An increase in the Hough threshold causes an increase in the average error for the low resolution. Similarly to that of the 1024x576 values, an increase of the Hough threshold increases the average error of the line prediction at 1280x960. At 1600x1200 an increased Hough threshold causes an increase in the average errors. This was consistent for all resolutions

Figure C.8 in Appendix C plots the number of angles detected for each Hough threshold at the three resolutions. Higher Hough thresholds were not always able to detect lines so that the camera could estimate its orientation. The higher the Hough threshold, the less likely the camera was to detect lines. This was especially true for further images, as that was where the camera detected the least amount of lines. The results of the effects of resolution, distance and Hough thresholds are discussed in the following section.

4.3.7 Discussion of Results

From all the results, conclusions are drawn which predict the effects of resolution, distance, and Hough threshold on the accuracy of the line detection. It was first important to discuss the difference between the experimental and theoretical models and the difference in their results. Figure 4.13 showed that there was a substantial difference in the errors obtained when using the experimental and theoretical models. The experimental model had significantly lower errors compared to the theoretical model. This result was because the theoretical model assumes the camera was positioned perfectly on the front of the robot. The rotation and translation matrices for the theoretical model assume that the camera lies at exactly 10 degrees from the vertical plane, and that no rotation about the Z axis has occurred. In practice however, this was not the case. It was likely that the camera was slightly rotated about the Z axis, slightly off-centre and perhaps not orientated perfectly at 10 degrees. The experimental model takes these factors into account, whereas the theoretical model does not. The experimental model was clearly a more accurate representation of the real camera extrinsic parameters and will always provide better results than the theoretical model unless the camera was fitted precisely.

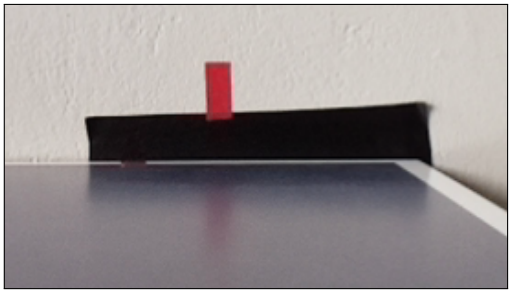
The fist parameter to influence the accuracy of the line detection was the resolution. In figures 4.14 and 4.15, it was seen that an increase in resolution lead to an increase in the accuracy of the line detection, regardless of the distance from the line. A higher resolution provides a higher degree of accuracy in the real-world co-ordinates, as the transformation map between image co-ordinates and world co-ordinates are more dense.

The next parameter worth discussing was that of the distance to the line. In Figure 4.16 and 4.17 it was clear that the average error for lines further away from the robot was higher than those closer to the robot. This was due to relationship produced by the mathematical models that were plotted in Figure 4.8b, which shows that the further away from the camera a point lies, the larger the change in Y between the points in that area. Using the fact that an angle calculation can be done using the ratio of delta Y and delta X with the tangential relationship, it was clear that a significant change in delta Y will change the angles obtained in real world co-ordinates. This explains why lines further away from the robot have larger degrees of error, and why the lines found closest to the robot are the most accurate.

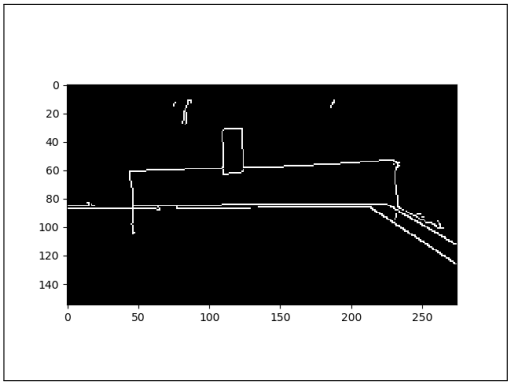
The last parameter to discuss was that of the Hough threshold. Figures 4.18 and C.8 displayed a trend of a higher Hough threshold leading to a higher average error for the lines detected. It was also seen that a higher Hough threshold meant that it was less likely that a line would be detected, especially at the further distances. This higher error with increasing Hough threshold are because at higher thresholds, the amount of lines found are significantly less than that of the lower thresholds. This means that the camera has less options to choose

in terms of finding the closest line to the actual line orientation as discussed in section 4.3.2. A lower threshold returns significantly more lines, meaning the system has more options when choosing the closest line to the known line orientation. It was also seen that a higher threshold finds significantly less lines overall, for some images, no lines were found. This means that the robot was unable to find the edge and would result in situations where no data was obtained. This was prominent at higher distances from the line.

Now that the effects of resolution, distance and Hough threshold have been considered, it was determined that the inaccuracies found in the system are not only caused by these factors, and other possible factors must be considered. The system was not always able to obtain lines exactly where the lines lie. To explain this, one needs to understand how Hough line detection works. Before Hough lines was performed, canny edge detection was used to extract all the edges based on the pixel intensities of the image. Canny edge detection has its own thresholds, which were kept at 50 for the lower bound and 150 for the higher bound during testing. These thresholds ensure that a point to be considered as part of an edge, must have a intensity gradient in any direction that lies between these thresholds. The problem that this introduces are that often false edges can be detected. Consider a point on an image that goes from black to white in a small region. The pixels that cover this shift from black to while will be a series of greys. If the image was taken very close to the black and white gradient, it was easier for the pixels to assign a clear grey value to themselves. If the image was taken further away from the black to white area, pixels may lie over a space with a high change in grey, meaning the pixel will choose some value for the grey in that pixel. This causes a blending of the pixels, especially when further away from the point in question. This blending makes it harder for the canny edge detection to determine what was a clear edge and what was not, which can lead to false edges being detected. An example of this is shown below in Figure 4.19, where the edge of the white line in Figure 4.19a has a glow effect around it. This was most likely to occur when the image was taken far away from the point of interest, or when the camera was unable to focus on the point of interest.



(a) Edge Blurring



(b) Result of Edge Blurring

Figure 4.19: The effect of pixel blurring on edge detection

The results of the edge blurring can be seen in Figure 4.19b, where there are multiple edges detected for one line, the edges detected are not co-linear or coincident. The edge detection has a knock-on effect to the Hough line detection, as the Hough line detection makes direct use of the resulting images of the Canny edge detection. If the Canny edge detection were returning the exact edges, the Hough lines would have no problem identifying the real lines in the image.

Chapter 5

Testing

Testing was done at the University of Stellenbosch's heliostat farm, which consists of a small power tower CSP setup with many heliostats in the field. The robot was tested for its reliability and cleaning speed. Surface coverage was unable to be tested as all forms of testing this could have damaged the mirror surface.

5.1 Testing Setup

Once the robot had been programmed and the accuracy of the camera had been determined, the robot was ready to be tested on a heliostat to determine its capability at both navigating the heliostat as well as the reliability of detecting the edge in real operating conditions. The robot was placed on a heliostat in the middle of a heliopod, with red edges fitted to the heliostat to allow the robot to identify the lines required for orientation determination. The testing setup with the robot on the mirror can be seen in figure 5.1.

The testing aims to provide a measure of the reliability of the robot and the speed at which it was capable of cleaning. The reliability of the robot was determined by its ability to clean without failing, to test this, the robot could navigate the mirror multiple times, with any failure scenarios noted. These are locations on the mirror where the robot may fail navigation, problems with the hardware that may prove restrictive or practical limitations that may be present. Testing the amount of surface area, the robot had cleaned was done by allowing it to navigate the mirror, and then noting areas which it did not reach after reaching the opposite end of the heliostat. The initial idea was to cover the glass with a layer of fine dirt to see the results, however this may have damaged the surface of the mirror, and as a result the cleaning area was instead estimated from knowing the movement of the robot during the cleaning algorithm. The cleaning speed was measured over various cleaning runs to get an idea of the speed the robot was capable of cleaning.

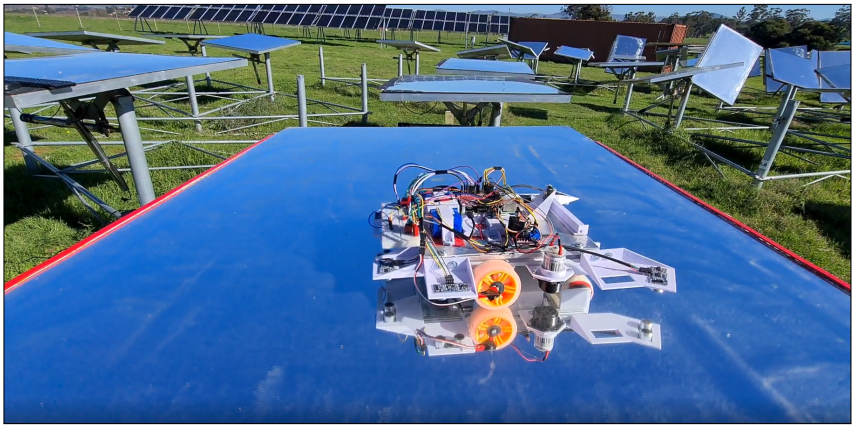


Figure 5.1: Testing setup with the robot on a heliostat

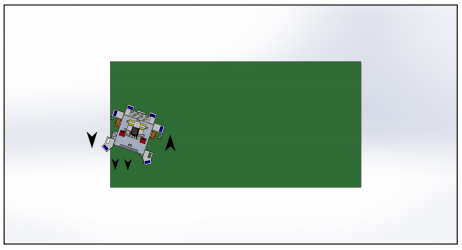
5.2 Testing Results

The robot was allowed to navigate over the mirror a number of times at different times of day and starting positions to determine the reliability of the system in real operating conditions. It was evident that when the camera was facing the direction of the sun the line detection became unreliable. This was due to sunlight reflecting into the camera lens and causing the red edge to appear lighter in colour. Although the line was painted in a non-reflective red, sunlight was causing it to appear as more of a white colour. The camera was set to detect lines in the red colour space. This was troublesome for the line detection, as the line it was searching for was no longer well defined, causing either false line detection or no lines detected. Another issue appeared to be the reflection of the red line on the mirror from the camera's perspective. The reflection of the line in the mirror led to additional false lines being detected, causing the orientation determination to be inaccurate. These camera restrictions meant that the robot was not able to rely on finding the red line alone to navigate the mirror. To counteract this, the ultrasonic sensors were used when the robot reached the edges to square the robot up with the edges using the front or rear sensors. This allowed the robot to orientate the robot to the edges, so that it was able to drive straight across the mirror without relying on the camera to find the line.

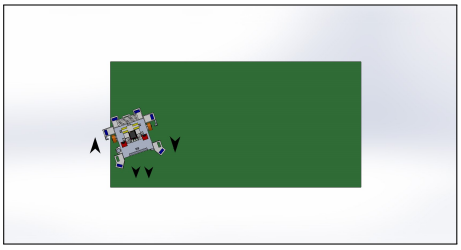
In addition to the camera unreliably finding the line, it was also worth noting additional problems the robot had when navigating the mirror. When using the ultrasonic sensors to square up to an edge, it was observed that using the back sensors took significantly longer than using the front sensors. This was due to the front sensors being placed wider than the rear sensors. The wider sensor positions allowed for quicker and more accurate squaring up to an edge. The robot also had navigation issues where the robot would incorrectly detect the corner after following the first edge. When the robot reaches a corner, it was required to perform a one hundred and eighty degree turn to

the left, however, it often occurred that the ultrasonic sensor on the front left did not detect the front edge of the mirror corner soon enough, causing it to fail navigation.

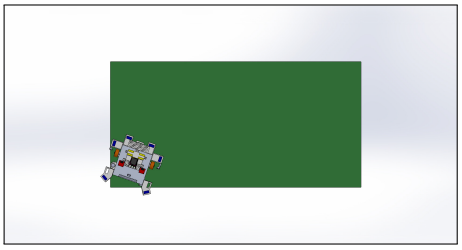
To understand why this occurs, one needs to understand the edge following process. Figure 5.2 provides the edge cleaning algorithm. When the front left sensor was over the edge as in figure 5.2a, the robot wants to turn left while driving forward. When the front left sensor was over the mirror, as in figure 5.2b, the robot wants to turn right while driving forward. When both sensors are off the edge it means that the robot was at the corner and can turn around, however it was likely that it could reach the corner without the front left sensor going over the edge, as in figure 5.2d. This meant the robot would continue trying to turn left even though it's at the corner, causing the right wheel to come off the mirror. Figure 5.2c shows the position required to move the robot to the next state. The solution to this was to add a single ultrasonic sensor in the front middle of the robot, positioned more forward than the current two, allowing it to detect the front edge more reliably.



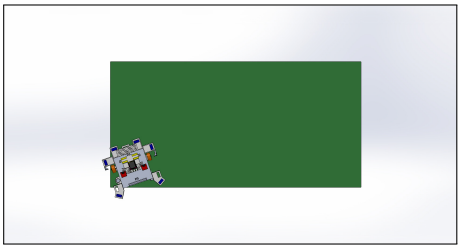
(a) Edge algorithm when sensor was off edge



(b) Edge algorithm when sensor was on edge



(c) Edge algorithm when sensor was off edge at corner

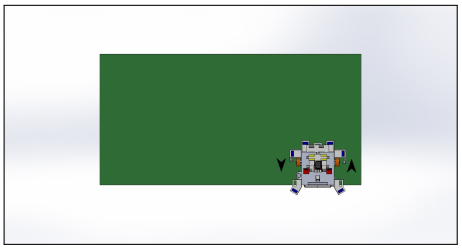


(d) Edge algorithm when sensor was on edge at corner

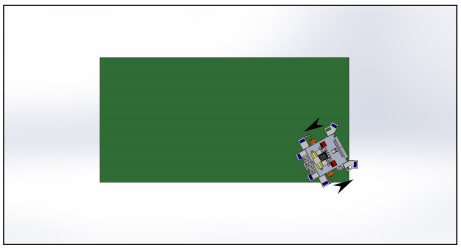
Figure 5.2: Edge cleaning algorithm

A similar issue could occur when following the last edge of the mirror, as the requirements to end the edge detection process are the same. Other than this, there was an additional problem in which the robot could fail its navigation. Once the robot had navigated the mirror and was making its final turn to follow the last edge, it could occur that the outer wheel came off the mirror. To understand why this can happen, one must also understand the

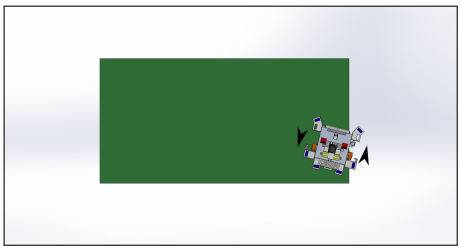
approach the robot takes to follow the final edge. When the robot reaches one of the edges, it executes a one hundred and eighty degree turn such as in figures 5.3a and 5.3b. The front sensor on the outside of the turn, in this case the front right sensor, was always looking for the side edge during the second ninety degrees of the turn. When it finds an edge, it begins the edge following algorithm. However, sometimes the robot begins the turn too close to the last edge, meaning the turning radius was too large to keep the wheels on the mirror such as in figures 5.3c and 5.3d. When a wheel comes off the mirror the robot navigation has failed. To solve this issue, a reverse command was issued to the robot when it detects the last edge, to narrow the turning circle and keep the wheel on the mirror.



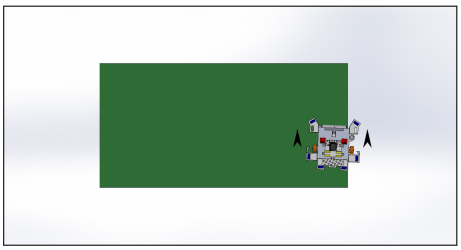
(a) Final turn to last Edge 1



(b) Final turn to last edge 2



(c) Final turn to last edge 3



(d) Final turn to last edge 4

Figure 5.3: Problem area with navigation 2

These were the two main problem areas experienced when testing the robot's navigation, but problems such as noise in the ultrasonic sensors and the sunlight as discussed previously also caused the robot to have issues navigating. Ultrasonic noise could be reduced by adding in a noise filter for each sensor or using higher quality sensors. The robot was ultimately capable of being placed anywhere in the centre of the mirror where it has space to rotate and find one of the red lines for the navigation process to start, but placing it near the sides may prove troublesome as it may not have enough space from the edges to rotate safely.

The amount of time the robot took to navigate the surface of the heliostat was varied by factors such as the speed in which it could find the red line, the time it took to square up to the edges, and the battery charge capacity. Table 5.1 shows the cleaning times for a series of runs done by the robot over

the heliostat. The heliostat measured 1.83 m by 1.22 m, meaning its surface area was a total of 2.2326 m^2. The average time to clean the heliostat was given as 08:10, meaning that the time per square meter was about 03:39.

Table 5.1: Time taken for the robot to clean a heliostat. The robot was capable of cleaning the robot in 04:43 after changes to the speed settings were made, meaning that there was room for improvement

Run Number	**Time**
1	07:39
2	08:02
3	07:14
4	09:46
Average	08:10

5.3 Requirement and Specification Analysis

In the design of the robot, section 3.1 and 3.2, requirements were established, and specifications were constructed from these requirements. This section briefly discusses whether the robot has met all these criteria. The robot was able to clean autonomously, with the only human interaction being its placement on and off the heliostat. Various shapes and sizes were unable to be tested, however the design conceptually allows for this, with some modification of the code likely required for alternating the shape. The robot was able to navigate safely for the most part, but problem areas have been highlighted in section 5.2. The robot was able to communicate wirelessly and provide water to its cleaning tool. The wheels were well under control with the selected motor drivers and the robot was able to make use of all its hardware. Data was processed in real time and ROS was installed on the system. It was able to recognise the heliostat edges, although not with complete reliability, and determine its orientation. The system was also powered by an on-board battery. The robot thus meets all the initial design requirements, albeit with a few shortfalls in terms of its reliability to detect and orientate itself to the line. Table 5.2 was provided which indicates which of the specifications have been met.

Table 5.2: Design specifications for the robot, compared to the achieved values

Specification	Required Value	Tested Value	Unit
Maximum Cleaning Time	5	03:39	minutes/m^2
Minimum Heliostat Size	2	2.2326	m^2
Maximum Edge Identification Time	10	5	seconds
Minimum Battery Life	2	6	hours
Maximum Mass	4	3.4	kg
Minimum Water Supply Size	500	500	ml
Maximum Time Between Sensor Data	200	200	milliseconds
Maximum Orientation Error	5	14	°

The only specification that was not met was that of the maximum orientation error. From the results in section 4.3.3, it was seen that the error was dependent on the distance, resolution and Hough threshold. The highest error obtained in the data was 13.6° off the known angle. This was found at the resolution of 1024x576. Other than this, all the other specifications have been met, with significant room for improvement allowing the error of the camera to be significantly reduced in the future.

Chapter 6

Conclusion

After testing the robot on a heliostat, it was clear that the robot has potential in cleaning heliostats commercially, but the concept needs further development before the system is reliable enough to be automated. The camera with Hough Lines is not yet a reliable means of detecting the edge, as the accuracy of the line detection does not provide reliable data, especially in outdoor environments. It was also shown that the experimental pose of the camera, a higher resolution, closer distance and lower Hough threshold provide better results for the line detection, although resolution does have a direct impact on the processing power of the system. It was likely that the accuracy of the camera can be significantly improved by using stereo vision or depth images. Alternatively, the edge detection algorithm can be trained using machine learning or AI to obtain better results. The ultrasonic sensors allowed the robot to detect and square up to edges when camera data was not available, but more are required to ensure that the robot can reliably progress between its navigation states. Despite these challenges, the robot was able to navigate the mirror using the navigation algorithm. A proof-of-concept robot was thus designed, built and tested that was capable of navigating a heliostat for the purpose of developing an automated cleaner, solving the need for high water use in conventional cleaning applications currently used on CSP plants.

6.1 Recommendations

Recommendations for the robot are discussed here, which highlight areas that can be improved for the next iteration of the concept.
The first recommendation for the system is to ensure that the cleaning roller is wider than the wheel positions of the robot. This will ensure that the cleaning area will be maximised, and that the cleaner will always overlap its previously travelled paths. The second recommendation is to obtain motors with encoders on the gearbox, not on the wheel. Currently, the encoders are on the wheels with a resolution of eight counts per rotation. Having them on the motor

gearbox will provide a much higher encoder resolution and thus more accurate control of the wheels. The next recommendation is to ensure that there are more ultrasonic sensors on the robot. More ultrasonic sensors will provide more data around the sides of the robot, making it detect edges and corners more reliably. The system should be designed to recharge itself with the use of a dock. Currently the battery and charge ports require manual connections to charge the system. It is recommended that stereo vision or depth cameras are tested for this application. If one makes use of the defined edge, such as the red line used in testing, it may be possible to obtain extremely accurate estimations of the robots pose on the mirror. The monocular camera was severely limited in its accuracy, especially when further away from the lines. GPS and IMU can also be added to the robot to provide more data about the robot's movement. It is recommended that the on-board computer be upgraded, or have the computation done on another device over a high speed network, as the image processing was limited by the capabilities of the CPU. Allowing higher frame rates on the camera ensures more data is available for the system.

6.2 The Way Forward

Now that the robot has been tested, it was clear that some aspects of the system would require further research and development to ensure the system was capable of reliably cleaning heliostats. The first factor that needs further research is that of the cleaning mechanism itself. Currently a roller with a small water supply is used, but this is unlikely to make any significant improvements to the cleanliness of a heliostat mirror. A cleaning process must thus be developed which can be implemented on the robot for heliostat cleaning. The second factor which needs further research is that of the camera accuracy. It was shown that the camera, although capable of finding the edges, was not accurate enough to control the movement of the robot. If a monocular camera is to be used, machine learning or artificial intelligence should be considered for detection of the line. Stereo vision can be tested on the heliostats, which may provide more accurate data compared to the monocular camera. A depth camera could also be tested, but its accuracy on the mirror surface may be inconsistent. The last factor that needs further improvement is that of the navigation algorithm. Currently, the system is unable to determine its position on the mirror if the navigation algorithm fails. A more robust algorithm should be developed which allows the robot to recover its position if it gets lost. The state navigation approach used is only reliable when accurate data is received from both the camera and ultrasonic sensors. Noise or inaccuracies from either of these can cause failures in the navigation.

References

[1] Solar PACES. *How CSP Works*. 2017. URL: https://www.solarpaces.org/csp-technologies/csp-how-it-works/.

[2] Rikki Allessandra. *Concentrated Solar Power vs. Photovoltaic*. 2019. URL: https://solarfeeds.com/csp-and-pv-differences-comparison/.

[3] Department of Energy USA Renewable Energy Office. *Power Tower System Concentrating Solar Power Basics | Department of Energy*. 2013. URL: https://www.energy.gov/eere/solar/articles/power-tower-system-concentrating-solar-powerbasics.

[4] R.S. Berg. "Heliostat dust buildup and cleaning studies". In: (1978).

[5] Michael Hardt et al. "Hector – Heliostat Cleaning Team-Orientated Robot". In: Nov. 2011.

[6] Herman K Trabish. *An Opportunity to Clean Up in Solar*. 2013. URL: https://www.greentechmedia.com/articles/read/an-opportunity-to-clean-up-in-solar#gs.sghwvr.

[7] CSP Focus. *Shouhang 100MW Concentrated Power Tower*. 2020. URL: http://heliosCSP.com/shouhang-bags-100-mw-concentrated-solar-power-tower-project-in-china/.

[8] ABC News. *Heliostat mirrors at a solar thermal power plant*. 2016. URL: https://www.abc.net.au/news/2016-09-08/heliostat-mirrors-at-a-solar-thermal-power-plant/7826308.

[9] Solar Energy Industries Association. *Concentrating Solar Power*. 2018. URL: https://www.seia.org/initiatives/concentrating-solar-power.

[10] Solar PACES. *CSP Projects Around the World*. 2017. URL: https://www.solarpaces.org/csp-technologies/csp-projects-around-the-world/.

[11] Helio CSP. *Environmental Impacts of CSP: Water, Land, Materials, Emissions, Flora and Fauna*. 2012. URL: http://heliosCSP.com/environmental-impacts-of-csp-water-land-materials-emissions-flora-and-fauna/.

[12] Nathan Bracken et al. *Concentrating Solar Power and Water Issues in the U.S. Southwest*. 2015.

[13] S.C Bhatia. *Solar Thermal Energy*. 2014.

[14] Solar Reserve. *Molten Salt Tower Reciever*. 2015. URL: https://www.solarreserve.com/en/technology/molten-salt-tower-receiver.

[15] Lauren Wood. *The Science behind Heliostats and Concentrated Solar Power (CSP) Heliostats*. 2013. URL: http://helioscsp.com/the-science-behind-heliostats-and-concentrated-solar-power-csp-heliostats/.

[16] Jeff Barbee. *South African team may have solved solar puzzle even Google couldnt crack*. 2015. URL: https://allianceearth.org/south-african-team-may-have-solved-solar-puzzle-even-google-couldnt-crack/.

[17] Francisco J. Collado. "Quick evaluation of the annual heliostat field efficiency". In: *Solar Energy* 82 (2008), pp. 379–384.

[18] MinWaterCSP. *New Cleaning Solutions*. 2018. URL: https://www.minwatercsp.eu/blog-31-new-cleaning-solutions-by-ecilimp_heliostats_parabolic-trough/.

[19] Andreas Pfahl et al. "Progress in heliostat development". In: *Solar Energy* (2017).

[20] Moishik Saraf. *SolaRobot Cleaner*. 2012. URL: http://solarobot-cleaner.com/.

[21] Christopher Sansom et al. "Contact cleaning of polymer film solar reflectors". In: AIP Conference Proceedings 1734. 2016.

[22] Open Source Robotics Foundation. *About ROS*. URL: https://www.ros.org/about-ros/.

[23] Yahya Tawil. *An Introduction to ROS*. 2017. URL: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/.

[24] Robinroy Peter. *ROS Topics*. 2019. URL: https://robinrobotic.blogspot.com/2019/07/ros-topics-ros-tutorial.html.

[25] Open Source Robotics Foundation. *ROS Introduction*. URL: http://wiki.ros.org/ROS/Introduction.

[26] Khan Saad Bin Hasan. *What, Why and How of ROS*. 2019. URL: https://towardsdatascience.com/what-why-and-how-of-ros-b2f5ea8be0f3.

[27] Robotics Backend. *What is ROS*. 2019. URL: https://roboticsbackend.com/what-is-ros/.

[28] MathWorks Team. *Edge Detection in Matlab and Simulink*. 2020. URL: https://www.mathworks.com/discovery/edge-detection.html.

[29] S.K. Katiyar and P.V. Arun. "Comparative analysis of common edge detection techniques in context of object extraction". In: *IEEE TGRS* 50 (2012), pp. 68–79.

[30] OpenCV Team. *OpenCV: Canny Edge Detection*. 2020. URL: https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html.

[31] OpenCV Team. *OpenCV: Smoothing Images*. 2020. URL: https://docs.opencv.org/trunk/d4/d13/tutorial_py_filtering.html.

[32] MathWorks Team. *Detecting Lines Using the Radon Transform in Matlab and Simulink*. 2020. URL: https://www.mathworks.com/help/images/detect-lines-using-the-radon-transform.html.

[33] Tomasz Praczyk. "A quick algorithm for horizon line detection in marine images". In: *J Mar Schi Technol* 23 (2018), pp. 164–177.

[34] Gideon Damaryam. "A Hough Transform Implementation for Line Detection for a Mobile Robot Self-Navigation System". In: *International Journal of Computer Engineering* 17 (Dec. 2015), pp. 2278–661. DOI: 10.9790/0661-17663344.

[35] C. Tu et al. "Vehicle Position Monitoring Using Hough Transform". In: *IERI Procedia* 4 (2013). 2013 International Conference on Electronic Engineering and Computer Science (EECS 2013), pp. 316–322. ISSN: 2212-6678. DOI: https://doi.org/10.1016/j.ieri.2013.11.045. URL: http://www.sciencedirect.com/science/article/pii/S2212667813000488.

[36] Socret Lee. *Lines Detection with Hough Transform*. 2020. URL: https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549.

[37] OpenCV Team. *OpenCV: Hough Line Transform*. 2020. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[38] J. G. Webster, S. Huang, and G. Dissanayake. "Robot Localization: An Introduction". In: *Wiley Encyclopedia of Electrical and Electronics Engineering* (2016).

[39] Comet Labs Research Team. *Teaching Robots Presence: What You Need to Know About SLAM*. 2017. URL: https://blog.cometlabs.io/teaching-robots-presence-what-you-need-to-know-about-slam-9bf0ca037553.

[40] R. Siegwart. *Autonomous Mobile Robots*. MIT Press, 2004.

[41] Faiza Gul, Wan Rahiman, and Syed Sahal Nazli Alhady. "A comprehensive study for robot navigation techniques". In: *Cogent Engineering* 6.1 (2019). Ed. by Kun Chen, p. 1632046. DOI: 10.1080/23311916.2019.1632046. eprint: https://www.tandfonline.com/doi/pdf/10.1080/23311916.2019.1632046. URL: https://www.tandfonline.com/doi/abs/10.1080/23311916.2019.1632046.

[42] T.J. Chong et al. "Sensor Technologies and Simultaneous Localization and Mapping". In: IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015). 2015.

[43] Danny Jost. *What is an Ultrasonic Sensor?* 2019. URL: https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor.

[44] Banner Engineering. *Clear and Reflective Targets*. 2020. URL: https://www.bannerengineering.com/us/en/solutions/smart-sensors/clear-and-reflective-targets.html.

[45] Rahul Kala. "2 - Basics of Autonomous Vehicles". In: *On-Road Intelligent Vehicles*. Ed. by Rahul Kala. Butterworth-Heinemann, 2016, pp. 11–35. ISBN: 978-0-12-803729-4. DOI: https://doi.org/10.1016/B978-0-12-803729-4.00002-7. URL: http://www.sciencedirect.com/science/article/pii/B9780128037294000027.

[46] Michal Dziergwa, Paweł Kaczmarek, and Jan Kędzierski. "RGB-D sensors in social robotics". In: *Journal of Automation Mobile Robotics and Intelligent Systems* 9 (Feb. 2015), pp. 18–27. DOI: 10.14313/JAMRIS_1-2015/3.

[47] Kenji Hata and Silvio Savarese. "CS231A Course Notes 1: Camera Models, Stanford University". In: (2020).

[48] Panfeng Huang et al. "Chapter 3 - Pose Measurement Based on Vision Perception". In: *Tethered Space Robot*. Ed. by Panfeng Huang et al. Academic Press, 2018, pp. 75–119. ISBN: 978-0-12-812309-6. DOI: https://doi.org/10.1016/B978-0-12-812309-6.00003-8. URL: http://www.sciencedirect.com/science/article/pii/B9780128123096000038.

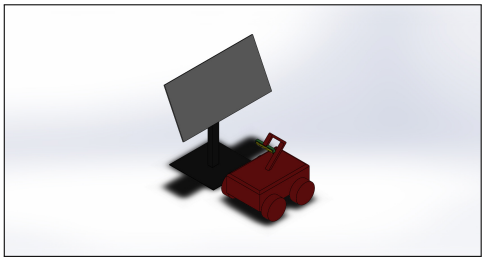
[49] Song Jingzhou and Caixiu Cao. "Pose Self-Measurement of Noncooperative Spacecraft Based on Solar Panel Triangle Structure". In: *Journal of Robotics* 2015 (Jan. 2015), pp. 1–6. DOI: 10.1155/2015/472461.

[50] Business Wire. *SolarReserve Teams with Heliostat SA to Create South Australian Jobs*. 2018. URL: https://www.businesswire.com/news/home/20180917005782/en/SolarReserve-Teams-Heliostat-SA-Create-South-Australian.

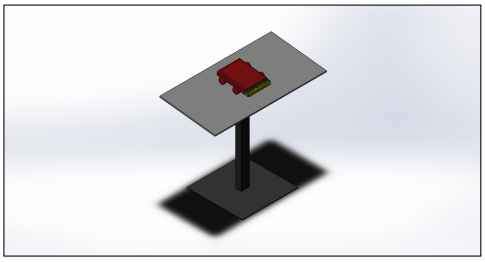
Appendices

Appendix A

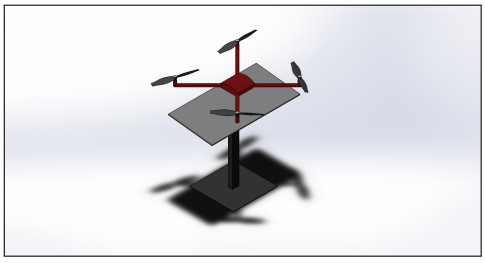
Additional Concepts



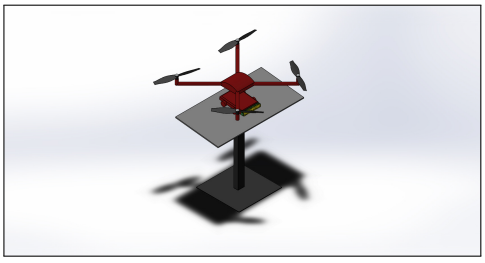
(a) Ground based cleaner



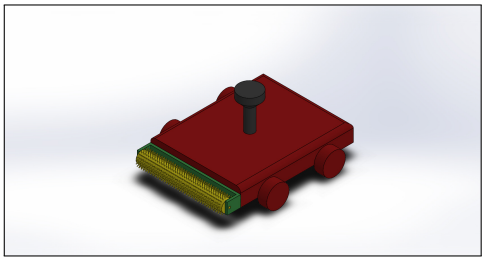
(b) Mirror based cleaning



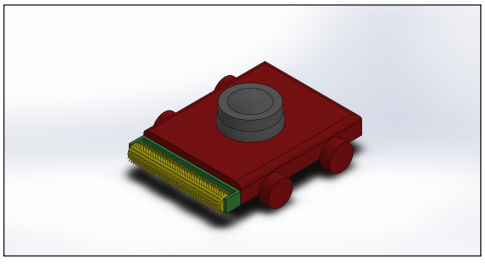
(c) Drone cleaner



(d) Drone with detachable cleaner



(e) Hemispherical radar sensor concept



(f) Hemispherical LIDAR sensor concept

Figure A.1: Additional concepts

Appendix B

Navigation Information and Pseudo Code

A PD controller in algorithm 2 is used to square the robot up to the edges using the camera. Once the robot is square with the line the PD control is ended, and the robot switches over to an alternate PD controller to drive straight across the mirror. If the line is not found within 8 iterations of the algorithm then the PD to line control is skipped. The PD controller controls the forward and reverse rotation of the right wheel.

A P controller in algorithm 5 was also developed which ensures the robot is able to drive straight using encoder data. This is important as the robot is not able to see the line all the time, and it allows the robot to navigate even when line data is not available. This code is given in algorithm 5.

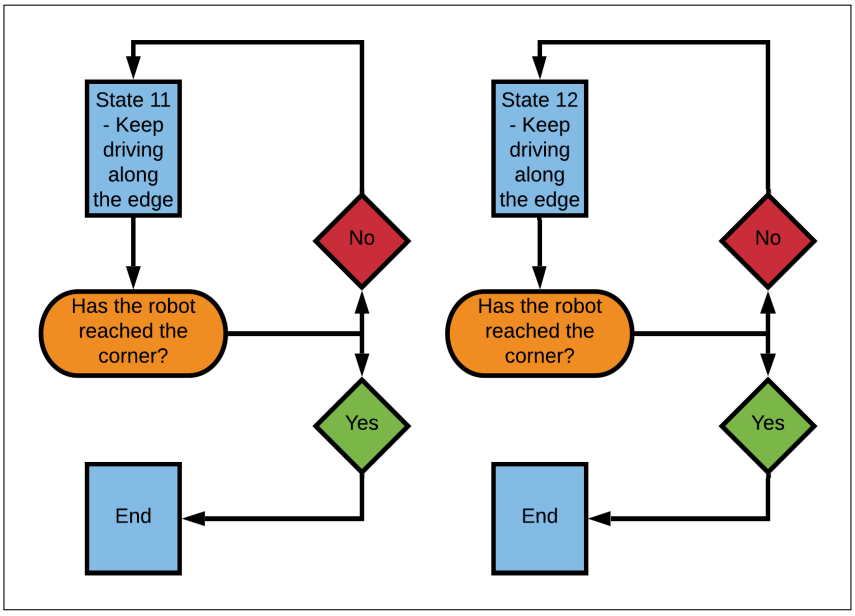


Figure B.1: Flow chart of states 11 and 12 for the robot

Algorithm 2 Proportional Derivative To Line

```
set all initial errors at zero and set the PD gains
Kp=2
Kd=1
while condition = 1 do
    read line gradient
    if gradient = 0 or count = 8 then
        condition = 0
        the robot is either square with the line or
        the robot was unable to find the line in 8 iterations
        the PD control is ended
    else if gradient = none then
        count = count+1
        we add to the counter, if we reach 8 we break the PD control
    else
        the terms for the PD are calculated
        set error = grad - 0
        set integral = integral prior + error * iteration time
        set derivative = (error-error prior) / iteration time
        calculate the output based on the error and derivative
        output = Kp * error + Kd * derivative
        update the error prior value for the next iteration
        set error prior = error
        ensure that the motor outputs are between 1 and -1
        if output ≥ 1 then
            output = 1
        else if output ≤ -1 then
            output = -1
        end if
        if output ≥ 0 then:
            set right wheel forward value to output*2
        else if output ≤ 0 then
            set right wheel reverse value to output*2
        end if
    end if
end while
```

Algorithm 3 Square Up Front Ultrasonic

```
1: while x = 0 do
2:     obtain latest front left and front right ultrasonic data
3:     if both front right and front left go low at the same time then
4:         the robot is square with the edge, set x = 1
5:     else if front right and front left is high then
6:         reverse the robot
7:     else if front right is high but front left is low then
8:         drive left wheel forward slowly
9:     else if front right is low but front left is high then
10:        drive right wheel forward slowly
11:    else if front right and front left is low then
12:        drive forward
13:    end if
14:    update the previous front left and front right value to the current value
15: end while
```

Algorithm 4 Edge Drive

```
1: obtain initial value for the front left ultrasonic.
2: while front left ultrasonic is low do
3:     obtain new values for the front right ultrasonic, right and front left
   ultrasonic
4:     if front left ultrasonic is high then
5:         break the loop
6:     else if front right ultrasonic and right is high then
7:         both right ultrasonic are off the edge
8:         drive forward left
9:     else if front right ultrasonic is high then
10:        the front right ultrasonic sensor is off the edge
11:        rotate slowly to the left
12:    else if front right ultrasonic is low then
13:        the front right ultrasonic sensor is on the mirror
14:        drive left wheel forward
15:    else
16:        stop
17:    end if
18: end while
```

Algorithm 5 Proportional Forward

```
obtain the initial values for the left and right encoders
set all initial errors at zero and set the gain
Kp=10
while PIDRun = 1 do
    obtain new left and right encoder values
    obtain new front ultrasonic values for edge detection
    if ultrasonic front right or ultrasonic left is high then
        PIDRun = 0 and break the loop
    end if
    set left difference = left value - initial left value
    set right difference = right value - initial right value
    set error = left difference - right difference, and output = Kp * error
    update the error prior value to the current error
    the output speed to the wheel is limited by the maximum speed defined
  by the robot
    if output ≥ RobotMaxSpeed then
        output = RobotMaxSpeed
    else if output ≤ -RobotMaxSpeed then
        output = -RobotMaxSpeed
    end if
    set left wheel value to base speed, set right wheel value to base speed
  + output
end while
```

The overall pseudo code for the navigation process is given in algorithm 6. This section of code is what controls the states of the robot, as well as the changing between the states.

Once the algorithm reaches state 3 it alternates between states 3 and 4 until it reaches the final edge. When the robot detects that it is at the final edge, it will initiate another edge following detection either on the left or right side depending on where the edge is. This is the final state of movement, and the cleaning process will be complete when the robot reaches the corner.

Algorithm 6 Navigation

```
1: while ROS is running do
2:     obtain latest gradient, ultrasonic values and encoder readings
3:     if True then
4:         the code is in state 0 by default, this is the initial state
5:         if state = 0 then
6:             if front right ultrasonic or front left ultrasonic is high then
7:                 the robot is at the front edge
8:                 square up the front of the robot to the edge using ultrasonics
9:                 change the state to state 100
10:            else if gradient = none and line has not been found then
11:                rotate the robot on the spot to find the line
12:            else if gradient = none but the line has been found then
13:                the robot drives forward with the proportional controller
14:            else
15:                the line is seen, the robot can drive toward it until it reaches
   the edge
16:            end if
17:        else if state = 100 then
18:            the robot performs a 90° turn left
19:            change the state to state 1
20:        else if state = 1 then
21:            if front right ultrasonic or front left ultrasonic is high then
22:                the robot is at the corner
23:                change the state to state 101
24:            else
25:                the robot drives forward with the proportional controller un-
   til it reaches the edge
26:                change the state to state 101
27:            end if
28:        else if state = 101 then
29:            square up the front of the robot to the edge at the corner
30:            the robot performs a 90° turn left
31:            change the state to state 2
32:        else if state = 2 then
33:            if front right ultrasonic or front left ultrasonic is high then
34:                the robot is at the next corner
35:                change the state to state 102
36:            else if gradient is seen then
37:                square up robot to observed line
38:                initiate the edge following algorithm to drive down the edge
39:                when edge algorithm is complete, change state to state 102
40:            else if gradient is not seen then
41:                skip the square up process
42:                initiate the edge following algorithm to drive down the edge
43:                when edge algorithm is complete, change state to state 102
44:            end if
```

Algorithm 6 Navigation (continued)

```
        else if state = 102 then
            perform a 180° turn left
            if turn occurs on the final edge of the mirror then
                follow the last edge
            end if
            search and square up to line with camera, if line isn't found then square up
            the back of the robot to edge with ultrasonics
            change the state to state 3
        else if state = 3 then
            if front right ultrasonic or front left ultrasonic is high then
                the robot is at the front edge
                change the state to 103
            else if gradient is seen then
                square up robot to observed line
                the robot drives forward with the proportional controller
                when the robot reaches the front edge, change state to 103
            else if gradient is not seen then
                skip the square up process
                the robot drives forward with the proportional controller
                when the robot reaches the front edge, change state to 103
            end if
        else if state = 103 then
            perform a 180° turn right
            if turn occurs on the final edge of the mirror then
                follow the last edge
            end if
            search and square up to line with camera, if line isn't found then square up
            the back of the robot to edge with ultrasonics
            change the state to state 4
        else if state = 4 then
            if front right ultrasonic or front left ultrasonic is high then
                the robot is at the front edge
                change the state back to 102
            else if gradient is seen then
                square up robot to observed line
                the robot drives forward with the proportional controller
                when the robot reaches the front edge, change state to 102
            else if gradient is not seen then
                skip the square up process
                the robot drives forward with the proportional controller
                when the robot reaches the front edge, change state to 102
            end if
        end if
    end if
end while
```

Appendix C

Additional Results

C.1 Additional Data for Expected Accuracy

Table C.1: World co-ordinates,(X, Y), and image co-ordinates,(u, v), of all resolutions for solvepnp

World X	World Y	576p u	576p v	960p u	960p v	1200p u	1200p v
46.5	196	672	476	840	714	1050	891
62.4	292	666	378	832	592	1039	739
-71.7	329	359	355	450	562	561	703
-78.6	195	251	480	313	718	392	898
104.3	224	833	442	1041	672	1300	840
36.1	396	581	321	724	520	904	650
-119.3	296	202	399	253	618	316	772
-123	392	282	323	352	523	442	654

Table C.2: Predicted world co-ordinates for theoretical and experimental pose at 1024x576

1024x576									
World Points		Theoretical Prediction				Experimental Prediction			
X	Y	X	Y	X Error	Y Error	X	Y	X Error	Y Error
46,5	196,0	45,8	241,8	1,5	23,4	46,3	193,4	0,5	1,3
62,4	292,0	62,4	345,7	0,0	18,4	63,4	291,5	1,6	0,2
-71,7	329,0	-80,4	402,3	12,2	22,3	-70,7	334,8	1,3	1,8
-78,6	195,0	-81,1	239,5	3,2	22,8	-78,8	195,8	0,2	0,4
104,3	224,0	104,7	272,4	0,4	21,6	104,1	218,0	0,2	2,7
36,1	396,0	33,5	457,9	7,2	15,6	37,8	397,5	4,7	0,4
-119,3	296,0	-125,8	320,1	5,5	8,1	-121,0	273,4	1,4	7,6
-123,0	392,0	-132,8	455,3	8,0	16,1	-124,1	402,9	0,9	2,8
			Average	4,7	18,5			1,3	2,1
			Max	12,2	23,4			4,7	7,6

Table C.3: Predicted world co-ordinates for theoretical and experimental pose at 1280x960

1280x960									
World Points		Theoretical Prediction				Experimental Prediction			
X	Y	X	Y	X Error	Y Error	X	Y	X Error	Y Error
46,5	196,0	46,5	233,2	0,1	19,0	46,2	194,2	0,7	0,9
62,4	292,0	62,8	332,3	0,6	13,8	63,5	291,6	1,8	0,1
-71,7	329,0	-72,7	369,3	1,4	12,2	-70,4	336,1	1,9	2,1
-78,6	195,0	-78,6	231,4	0,0	18,7	-78,9	196,8	0,4	0,9
104,3	224,0	104,3	261,5	0,0	16,8	103,9	218,6	0,4	2,4
36,1	396,0	35,0	437,2	3,0	10,4	37,7	399,2	4,3	0,8
-119,3	296,0	-120,8	306,7	1,3	3,6	-120,6	273,1	1,1	7,7
-123,0	392,0	-125,7	433,3	2,2	10,5	-124,5	403,0	1,2	2,8
			Average	1,1	13,1			1,5	2,2
			Max	3,0	19,0			4,3	7,7

Table C.4: Predicted world co-ordinates for theoretical and experimental pose at 1600x1200

1600x1200									
World Points		Theoretical Prediction				Experimental Prediction			
X	Y	X	Y	X Error	Y Error	X	Y	X Error	Y Error
46,5	196,0	45,5	238,6	2,1	21,7	46,3	193,9	0,4	1,1
62,4	292,0	62,0	342,3	0,7	17,2	63,6	291,8	1,9	0,1
-71,7	329,0	-78,6	382,6	9,7	16,3	-70,8	335,2	1,3	1,9
-78,6	195,0	-82,3	237,1	4,7	21,6	-78,8	196,1	0,2	0,6
104,3	224,0	104,8	268,5	0,4	19,8	103,9	217,5	0,4	2,9
36,1	396,0	33,0	455,4	8,7	15,0	37,5	399,0	3,9	0,8
-119,3	296,0	-126,9	315,2	6,3	6,5	-120,8	273,2	1,2	7,7
-123,0	392,0	-134,4	450,4	9,3	14,9	-123,7	402,4	0,5	2,6
			Average	5,2	16,6			1,2	2,2
			Max	9,7	21,7			3,9	7,7

C.2 Camera Data

Camera matrix and distortion coefficients for 1024x576:

$$\begin{bmatrix} 827.3026550499602 & 0 & 520.8319796222933 \\ 0 & 828.2099978343125 & 285.6178278051194 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.1}$$

$$\begin{bmatrix} 0.1879566936302154 \\ -0.3266709238030808 \\ 0.004431216935420947 \\ 0.001661415201566401 \\ 0 \end{bmatrix} \tag{C.2}$$

Camera matrix and distortion coefficients for 1280x960:

$$\begin{bmatrix} 1019.556629428377 & 0 & 645.3611535215274 \\ 0 & 1020.688497776793 & 467.2202459617425 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.3}$$

$$\begin{bmatrix} 0.166163448802218 \\ -0.2758222580922626 \\ -0.0001541701860594309 \\ 0.0001924568580878326 \\ 0 \end{bmatrix} \tag{C.4}$$

Camera matrix and distortion coefficients for 1600x1200:

$$\begin{bmatrix} 1269.713433285016 & 0 & 817.187819153611 \\ 0 & 1272.496143005405 & 595.1769100548456 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.5}$$

$$\begin{bmatrix} 0.1789180466860549 \\ -0.2986651009388727 \\ 0.003445295197871999 \\ 0.002330990293923629 \\ 0 \end{bmatrix} \tag{C.6}$$

Experimental rotation and translation matrix for 1024x576:

$$\begin{bmatrix} 0.9997624 & -0.01930852 & -0.01011475 \\ -0.01349436 & -0.18384326 & -0.98286295 \\ 0.0171181 & 0.9827659 & -0.18406014 \end{bmatrix} \tag{C.7}$$

$$\begin{bmatrix} 2.34540311 \\ 92.64406019 \\ 54.83488667 \end{bmatrix} \tag{C.8}$$

Experimental rotation and translation matrix for 1280x960:

$$\begin{bmatrix} 0.99981284 & -0.01666929 & -0.00981846 \\ -0.01256565 & -0.17366171 & -0.9847252 \\ 0.01470958 & 0.98466426 & -0.17383868 \end{bmatrix} \tag{C.9}$$

$$\begin{bmatrix} 3.2033756 \\ 92.72687918 \\ 49.75226184 \end{bmatrix} \tag{C.10}$$

Experimental rotation and translation matrix for 1600x1200:

$$\begin{bmatrix} 0.9996434 & -0.02555772 & -0.00774187 \\ -0.01225796 & -0.18159008 & -0.9832979 \\ 0.02372501 & 0.9830421 & -0.1818386 \end{bmatrix} \tag{C.11}$$

$$\begin{bmatrix} 3.03470671 \\ 92.02388878 \\ 50.21834117 \end{bmatrix} \tag{C.12}$$

Theoretical Model for 1024x576

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 827.30 & 0 & 520.83 \\ 0 & 828.21 & 285.62 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -sin(10°) & -cos(10°) & 99.53 \\ 0 & cos(10°) & -sin(10°) & 17.55 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.13}$$

Theoretical Model for 1280x960

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1019.56 & 0 & 645.36 \\ 0 & 1020.69 & 467.22 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -sin(10°) & -cos(10°) & 99.53 \\ 0 & cos(10°) & -sin(10°) & 17.55 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.14}$$

Theoretical Model for 1600x1200

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1269.71 & 0 & 817.19 \\ 0 & 1272.50 & 595.18 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -sin(10°) & -cos(10°) & 99.53 \\ 0 & cos(10°) & -sin(10°) & 17.55 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.15}$$

Experimental Model for 1024x576

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 827.30 & 0 & 520.83 \\ 0 & 828.21 & 285.62 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1.00 & -0.02 & -0.01 & 2.35 \\ -0.01 & -0.18 & -0.98 & 92.64 \\ 0.02 & 0.98 & -0.18 & 54.83 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.16}$$

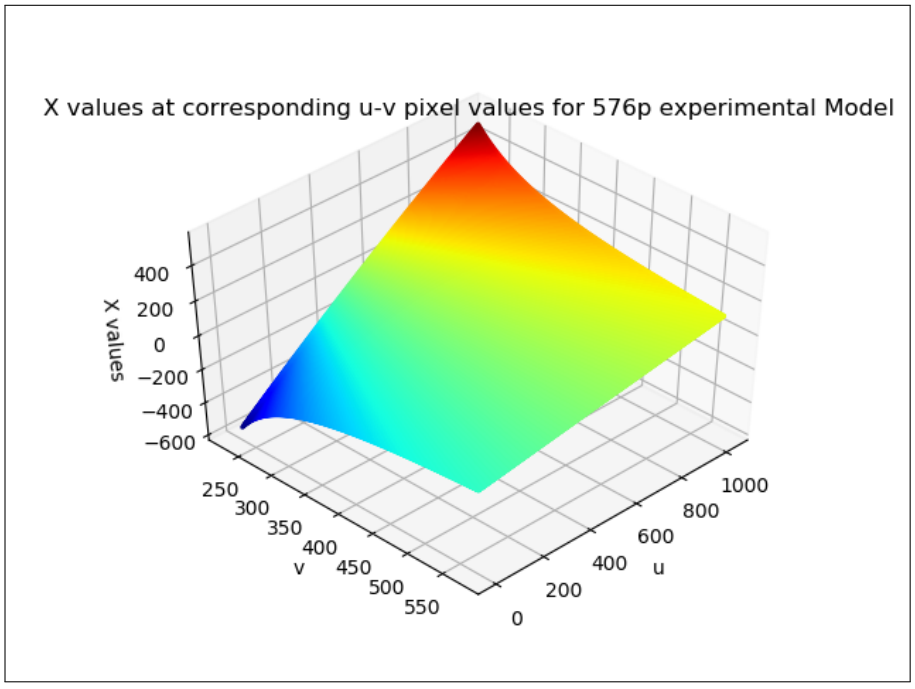
Experimental Model for 1280x960

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1019.56 & 0 & 645.36 \\ 0 & 1020.69 & 467.22 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1.00 & -0.02 & -0.01 & 3.20 \\ -0.01 & -0.17 & -0.98 & 92.73 \\ 0.015 & 0.98 & -0.17 & 49.75 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.17}$$

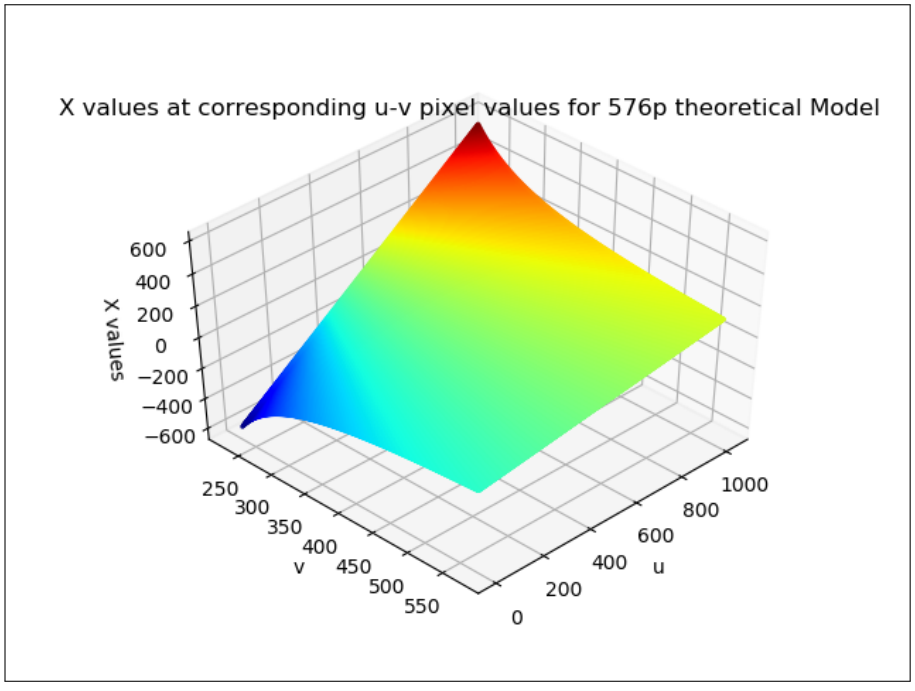
Experimental Model for 1600x1200

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1269.71 & 0 & 817.19 \\ 0 & 1272.50 & 595.18 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1.00 & -0.03 & -0.01 & 3.03 \\ -0.01 & -0.18 & -0.98 & 92.02 \\ 0.02 & 0.98 & -0.18 & 50.22 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{C.18}$$

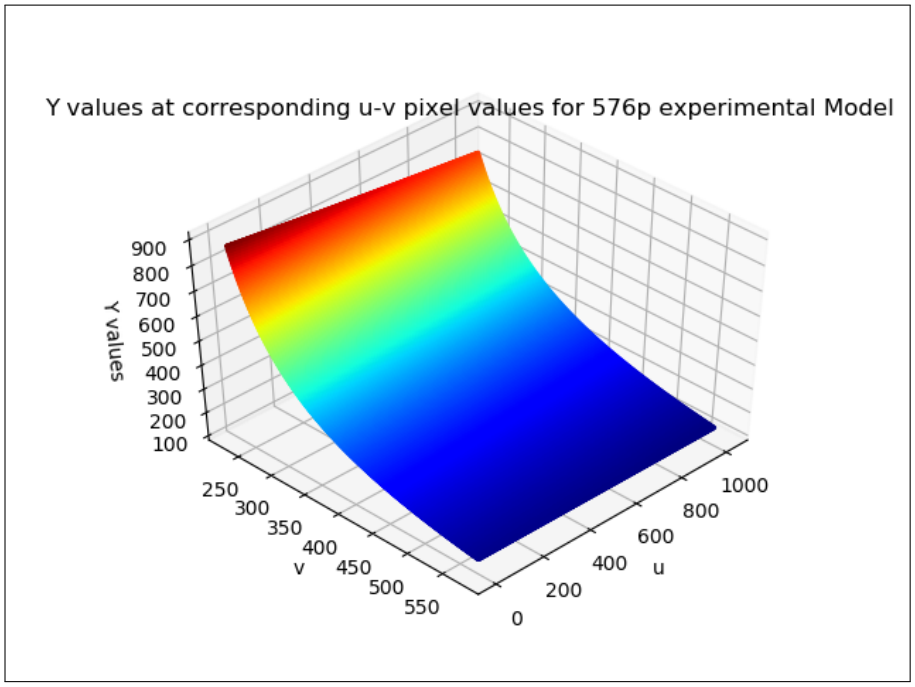
C.3 Additional Graphs



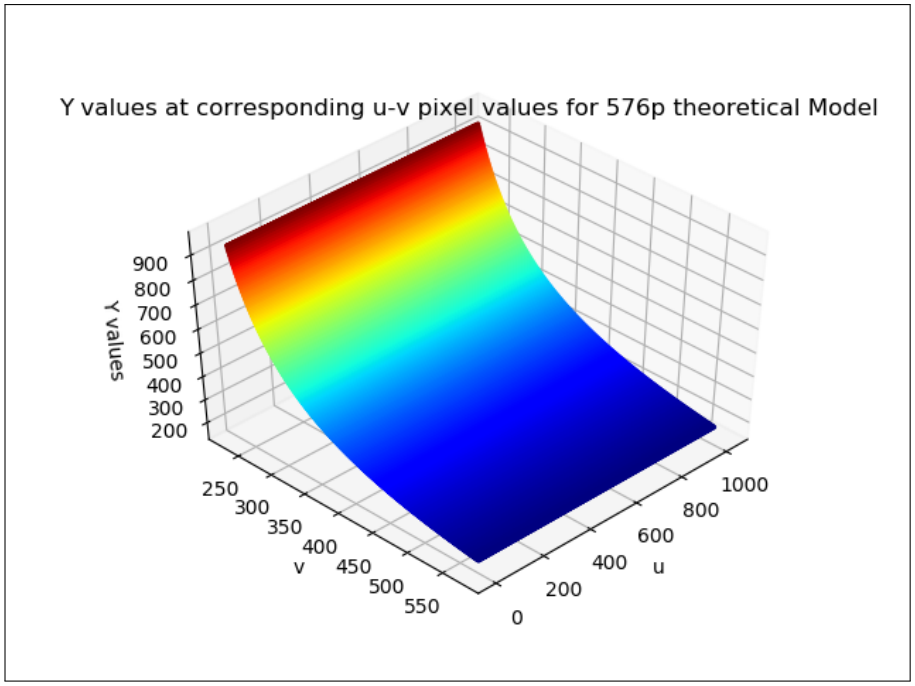
(a) X Values at corresponding pixel co-ordinates for the 1024x576 experimental model



(b) X values at corresponding pixel co-ordinates for the 1024x576 theoretical model

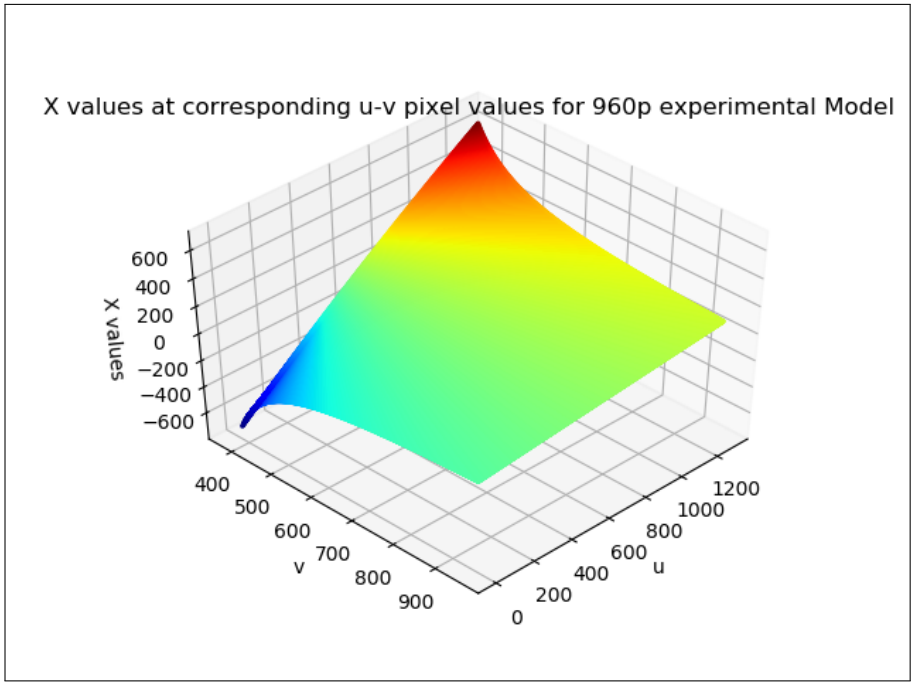


(c) Y values at corresponding pixel co-ordinates for the 1024x576 experimental model

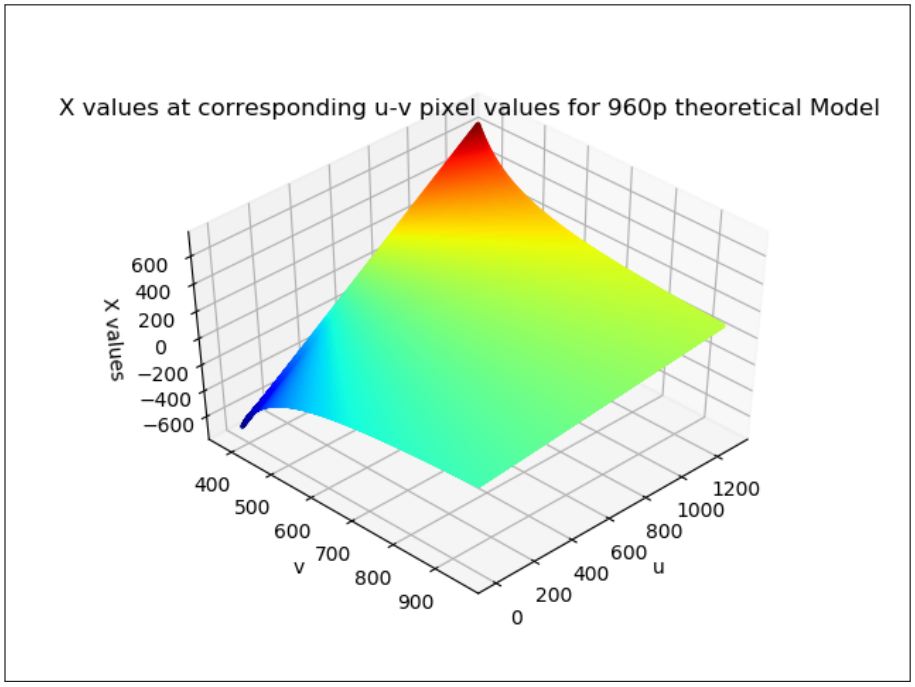


(d) Y values at corresponding pixel co-ordinates for the 1024x576 theoretical model

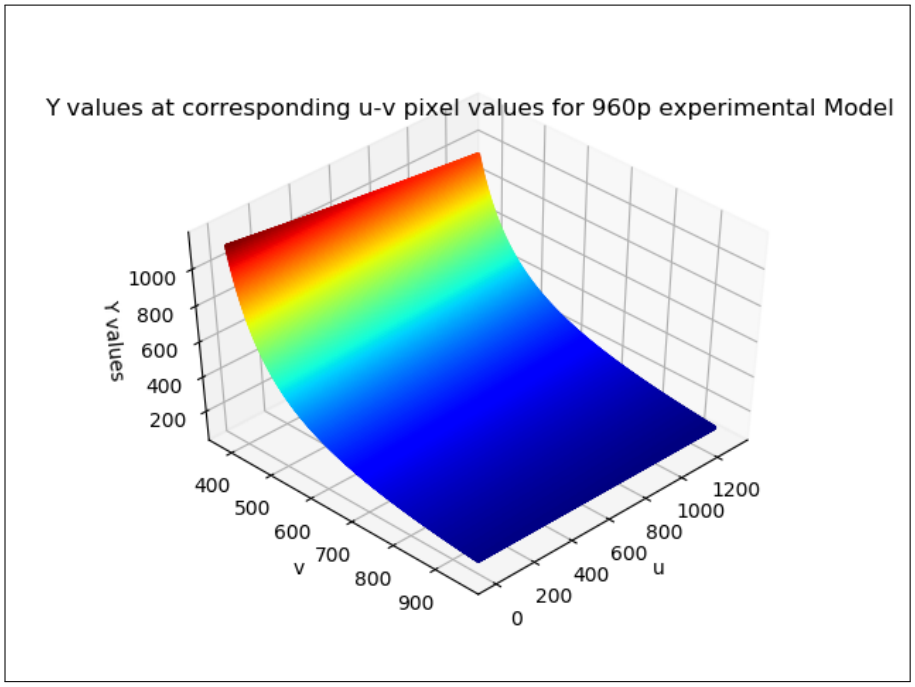
Figure C.1: X-Y plots for u-v values (1/3)



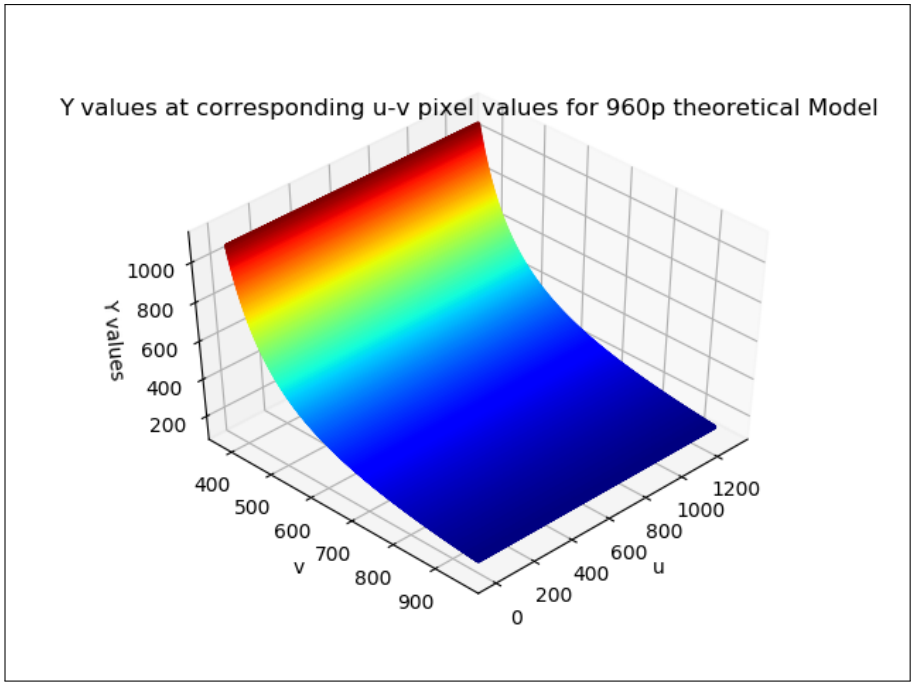
(a) X values at corresponding pixel co-ordinates for the 1280x960 experimental model



(b) X values at corresponding pixel co-ordinates for the 1280x960 theoretical model

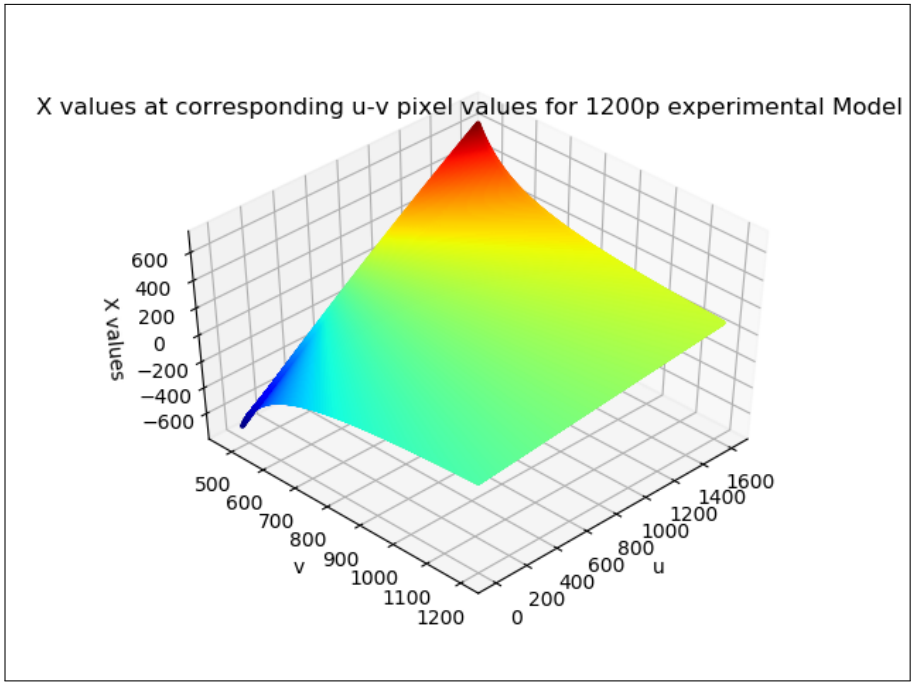


(c) Y values at corresponding pixel co-ordinates for the 1280x960 experimental model

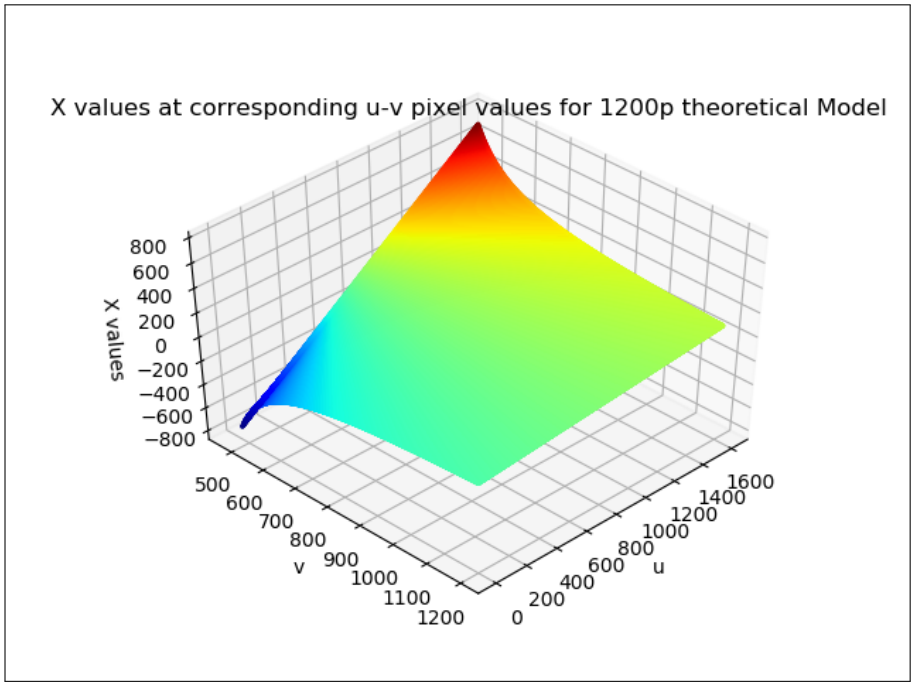


(d) Y values at corresponding pixel co-ordinates for the 1280x960 theoretical model

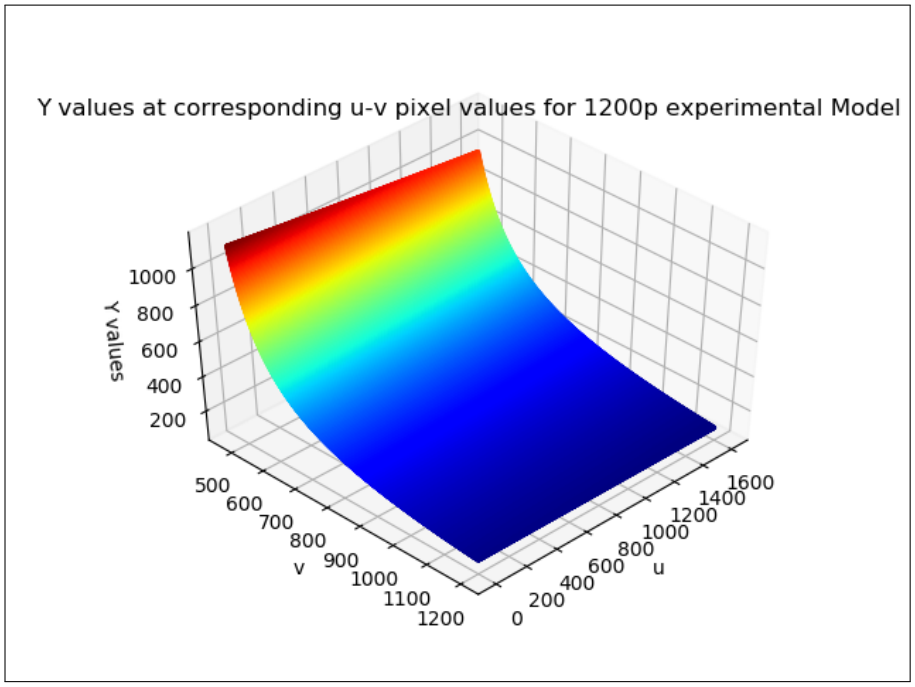
Figure C.2: X-Y plots for u-v values (2/3)



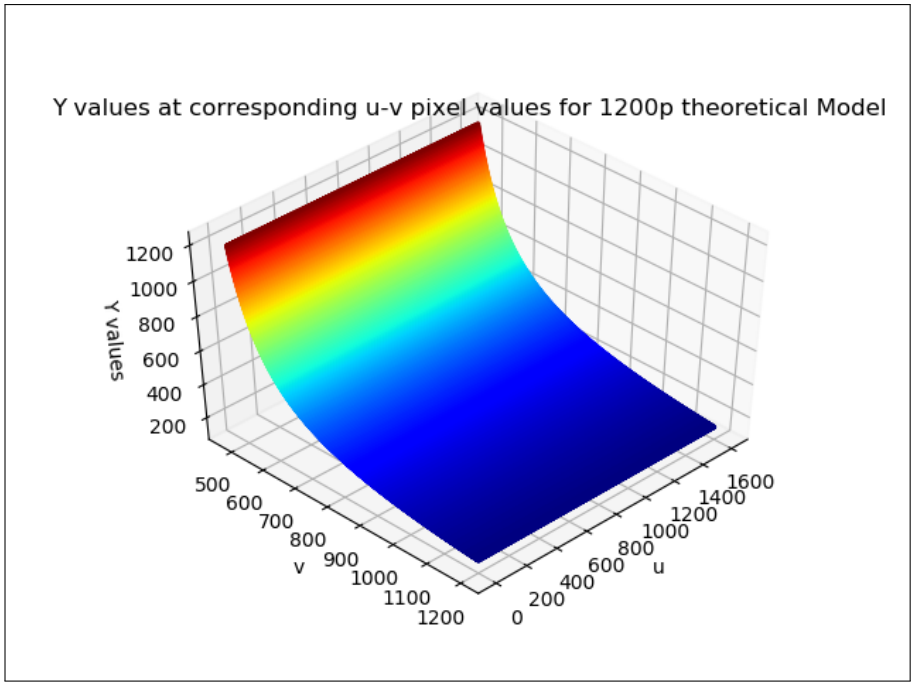
(a) X values at corresponding pixel co-ordinates for the 1600x1200 experimental model



(b) X values at corresponding pixel co-ordinates for the 1600x1200 theoretical model

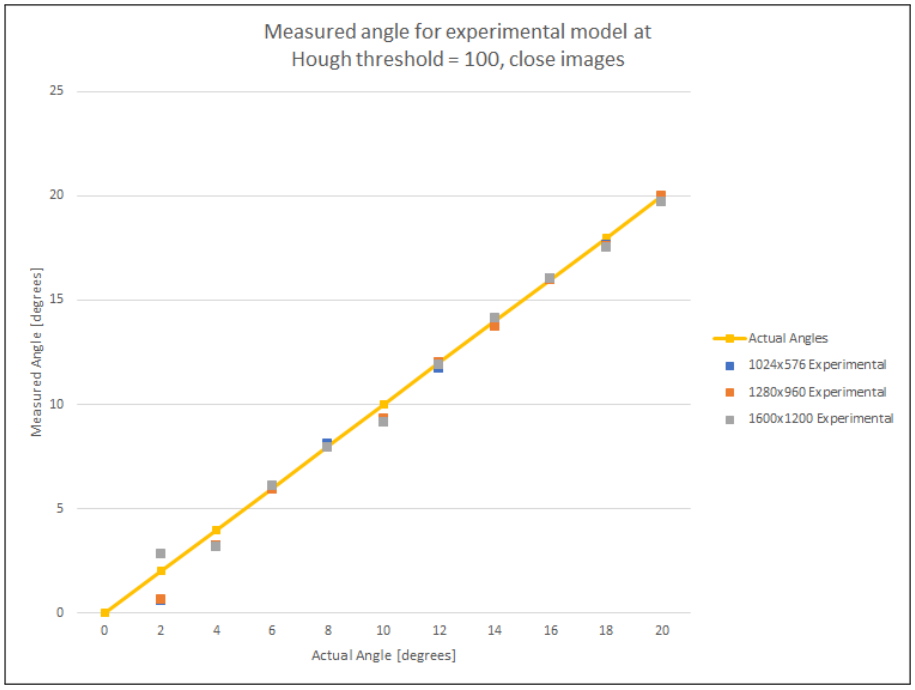


(c) Y values at corresponding pixel co-ordinates for the 1600x1200 experimental model

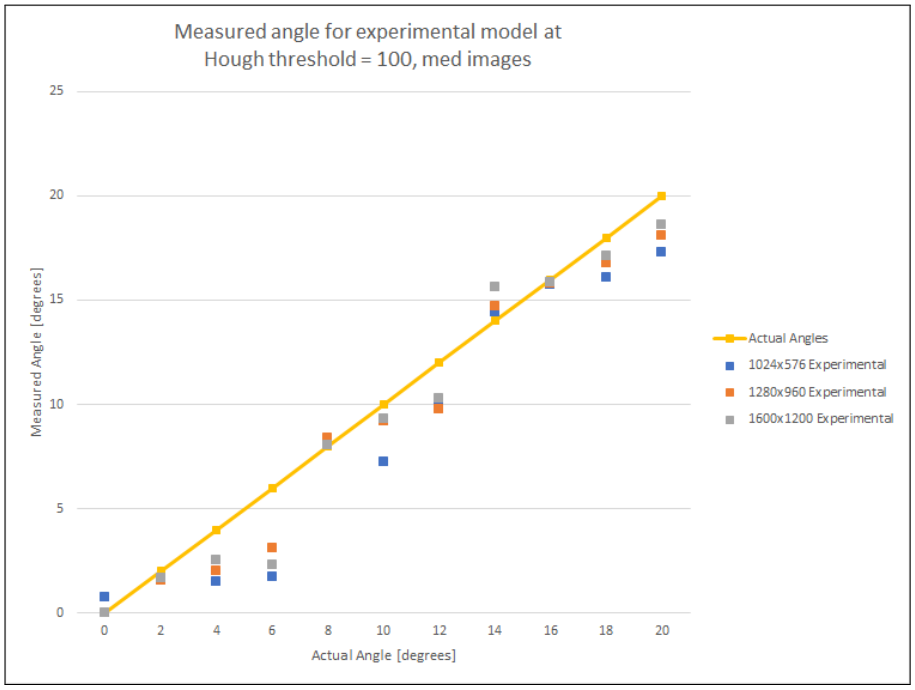


(d) Y values at corresponding pixel co-ordinates for the 1600x1200 theoretical model

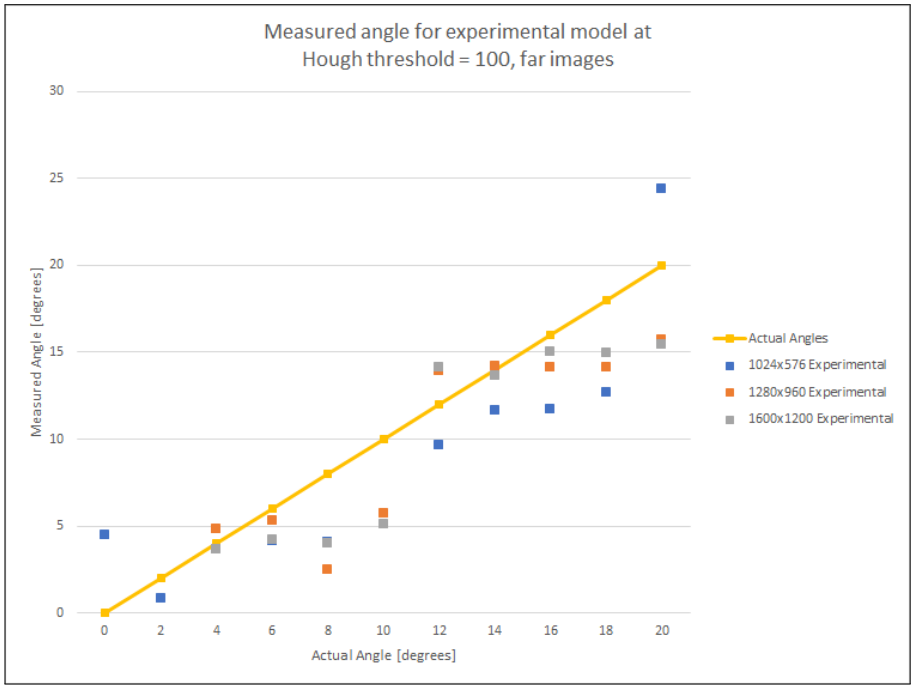
Figure C.3: X-Y plots for u-v values (3/3)



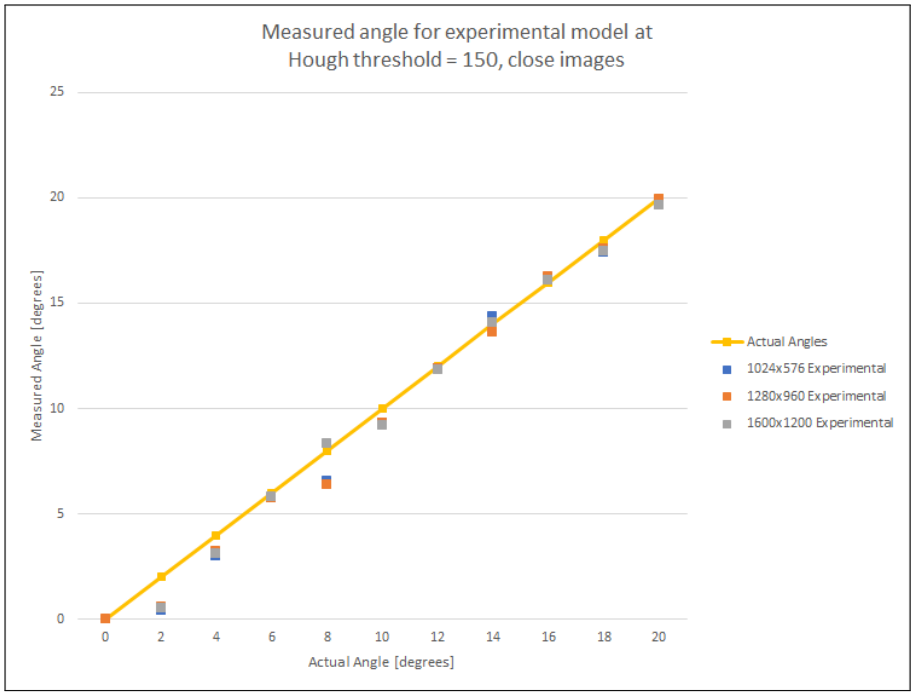
(a) Measured angles for experimental model at Hough threshold 100 for close images



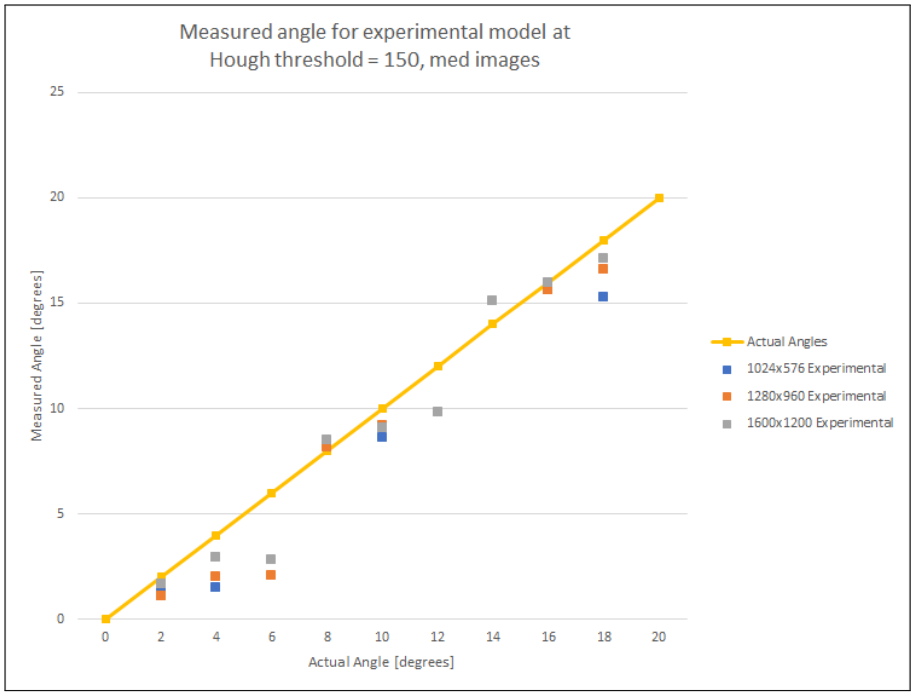
(b) Measured angles for experimental model at Hough threshold 100 for medium images



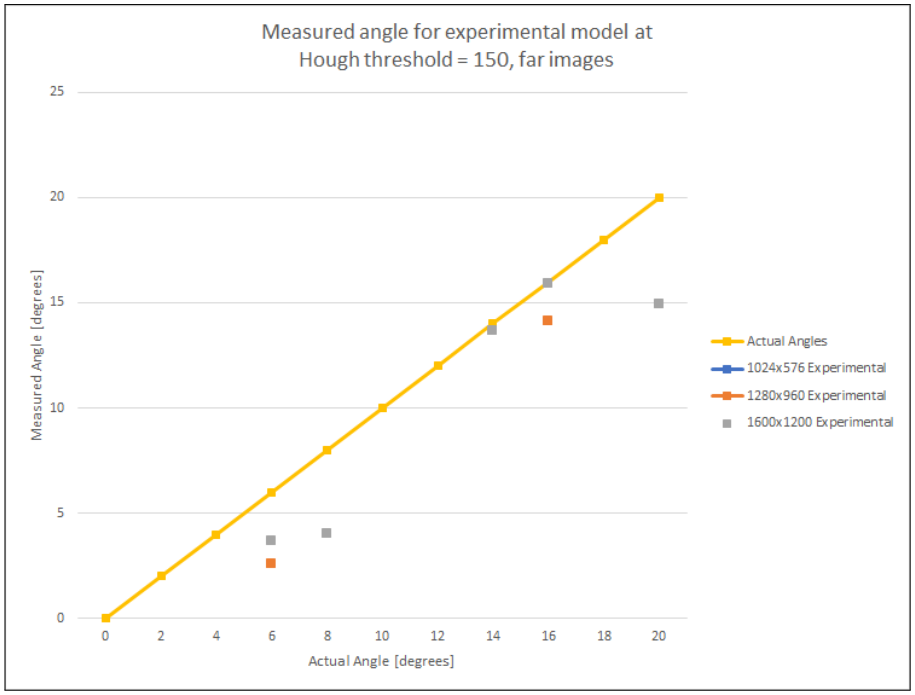
(c) Measured angles for experimental model at Hough threshold 100 for far images



(d) Measured angles for experimental model at Hough threshold 150 for close images

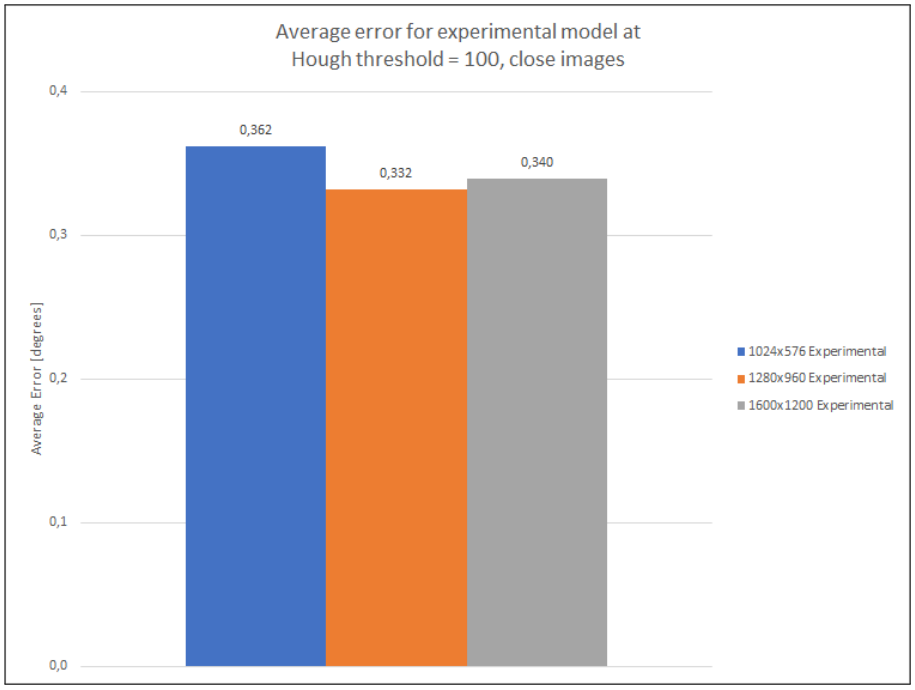


(e) Measured angles for experimental model at Hough threshold 150 for medium images

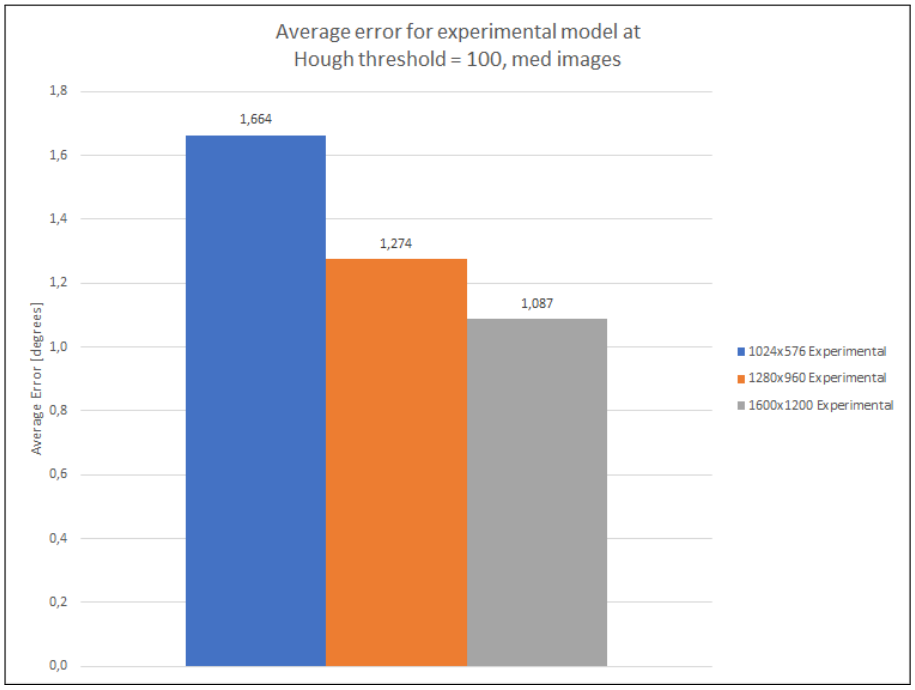


(f) Measured angles for experimental model at Hough threshold 150 for far images

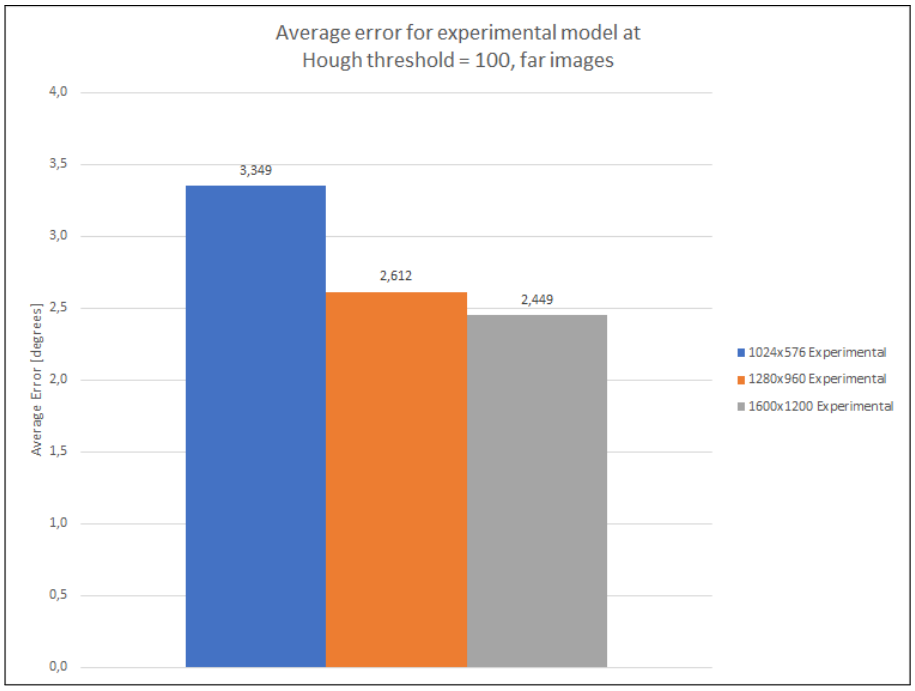
Figure C.4: Graphs of angle vs distance at threshold 100 and 150



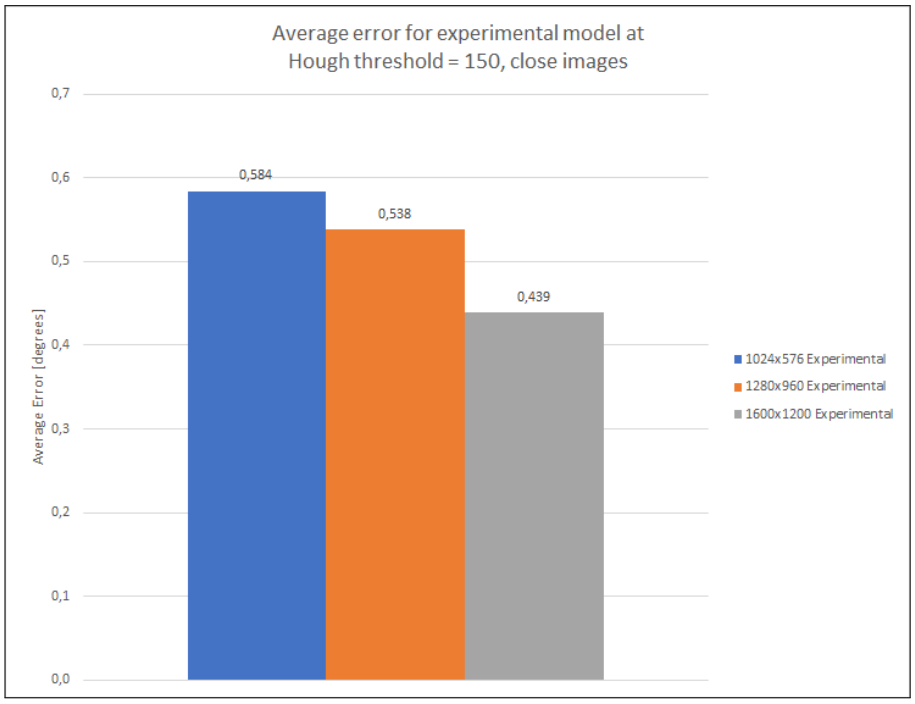
(a) Average error for various resolutions at Hough threshold 100, close images



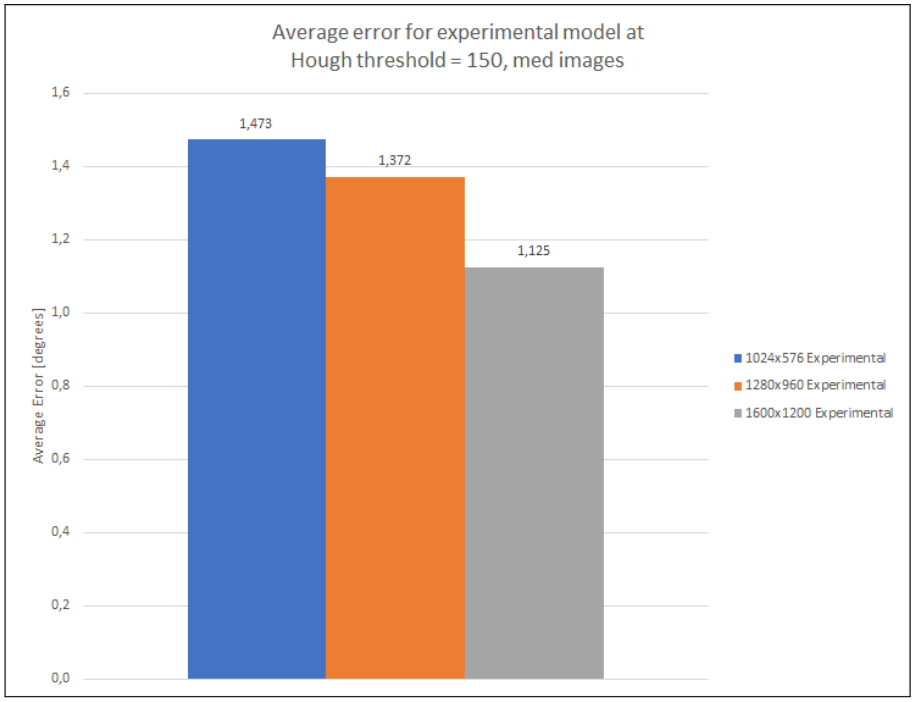
(b) Average error for various resolutions at Hough threshold 100, medium images



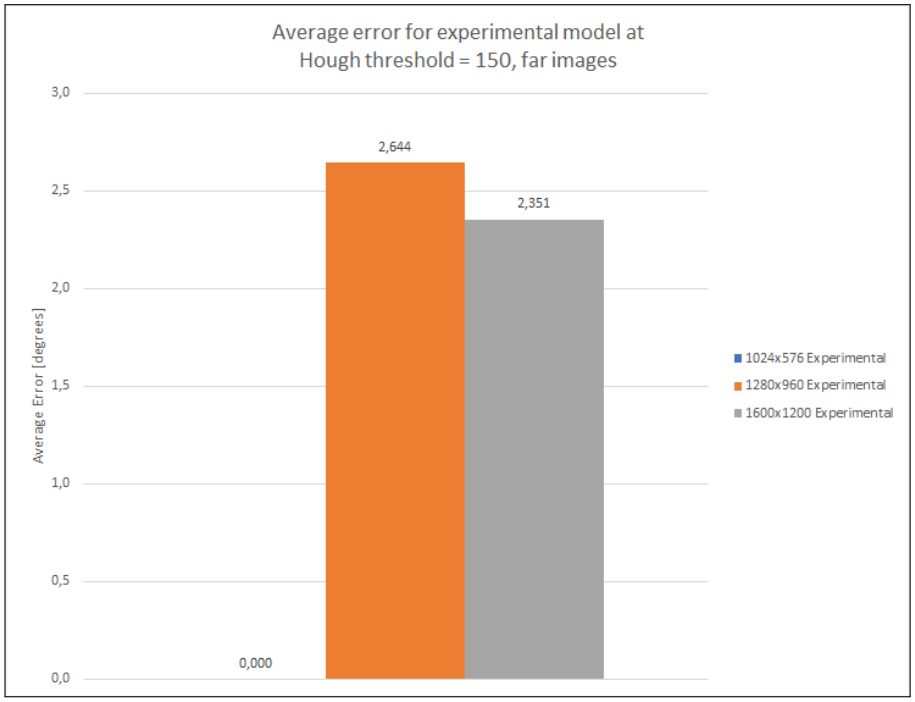
(c) Average error for various resolutions at Hough threshold 100, far images



(d) Average error for various resolutions at Hough threshold 150, close images

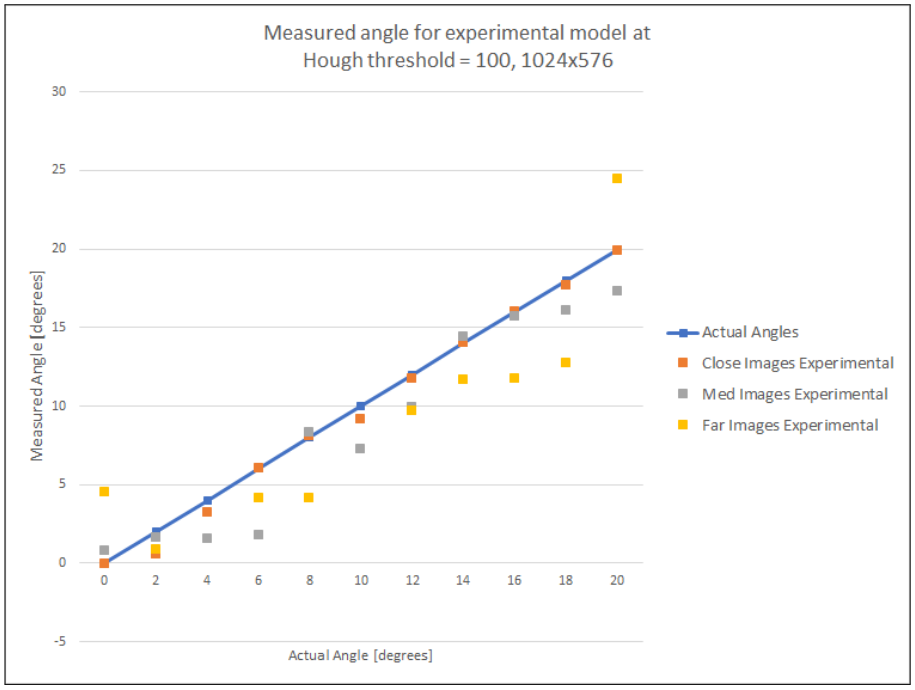


(e) Average error for various resolutions at Hough threshold 150, medium images

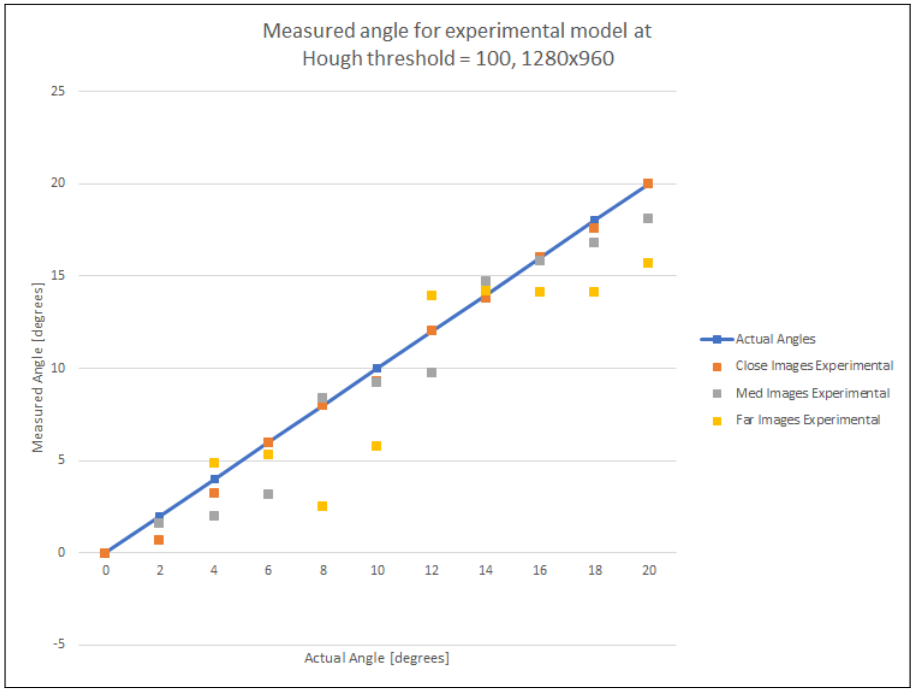


(f) Average error for various resolutions at Hough threshold 150, far images

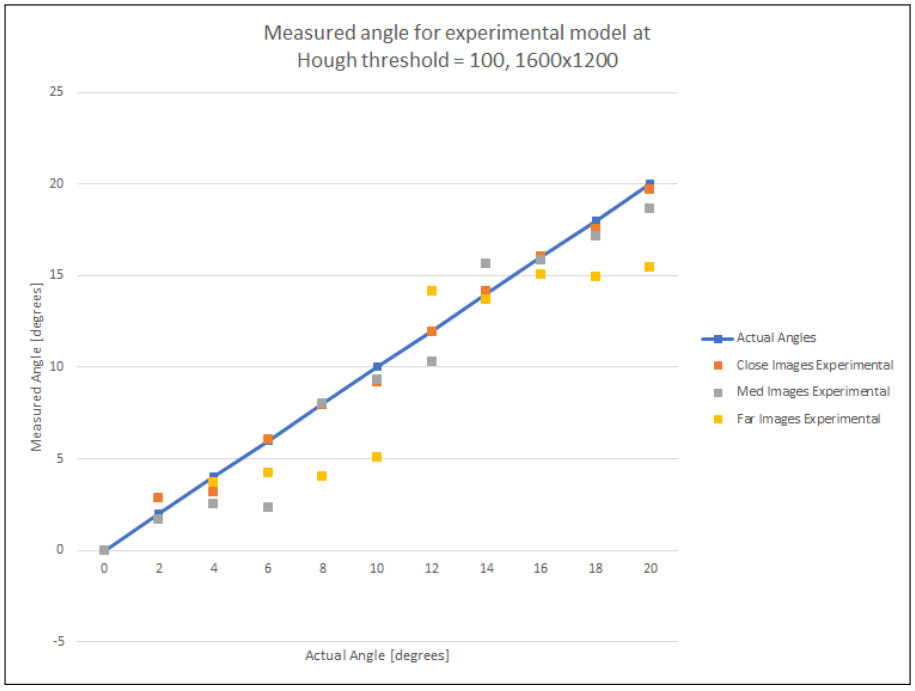
Figure C.5: Graphs of average error vs resolution at threshold 100 and 150



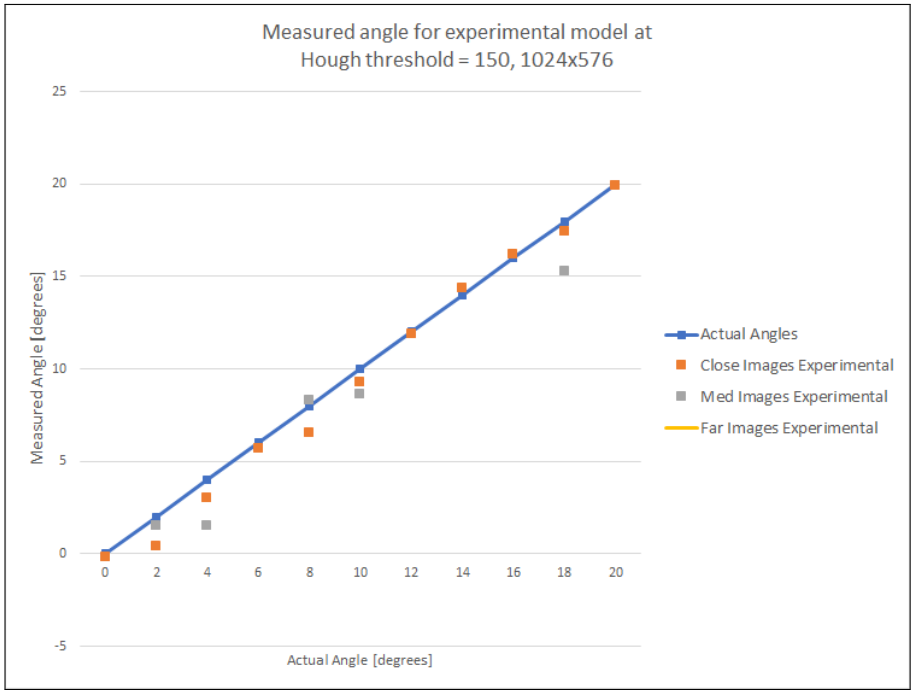
(a) Measured angles for experimental model at Hough threshold 100 for 1024x576



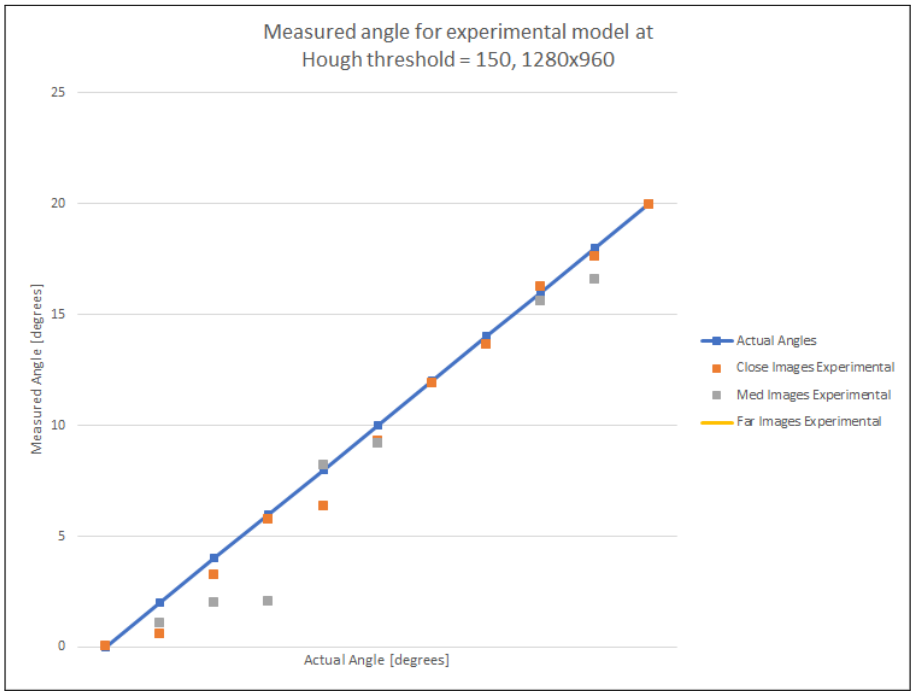
(b) Measured angles for experimental model at Hough threshold 100 for 1280x960



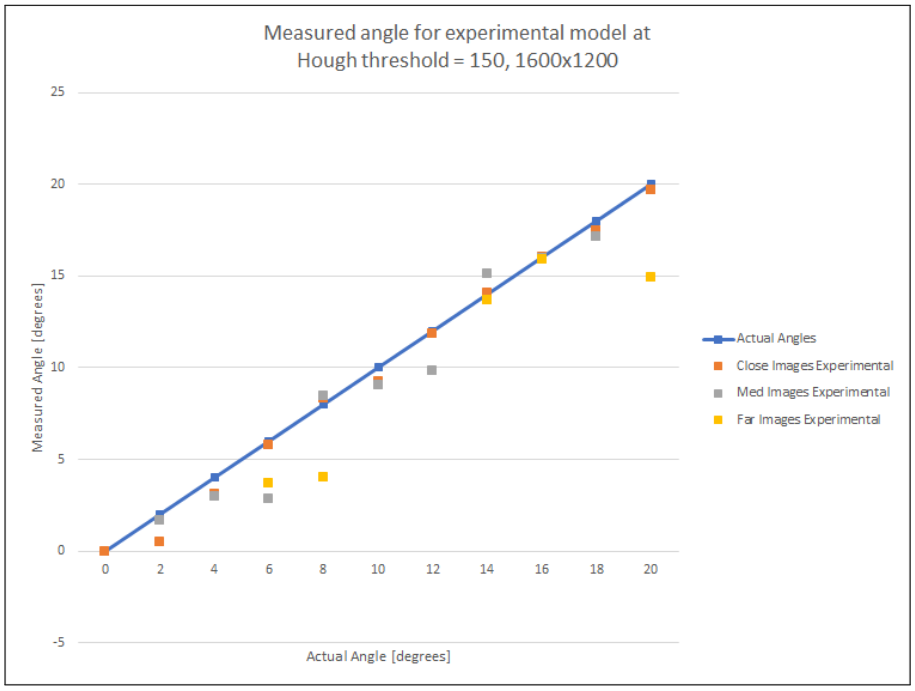
(c) Measured angles for experimental model at Hough threshold 100 for 1600x1200



(d) Measured angles for experimental model at Hough threshold 150 for 1024x576

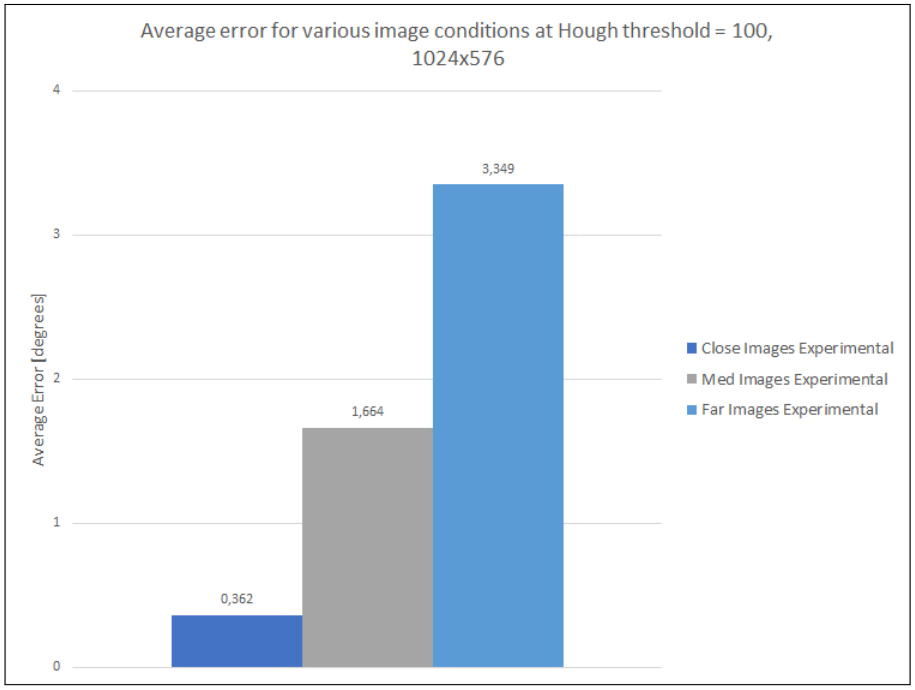


(e) Measured angles for experimental model at Hough threshold 150 for 1280x960

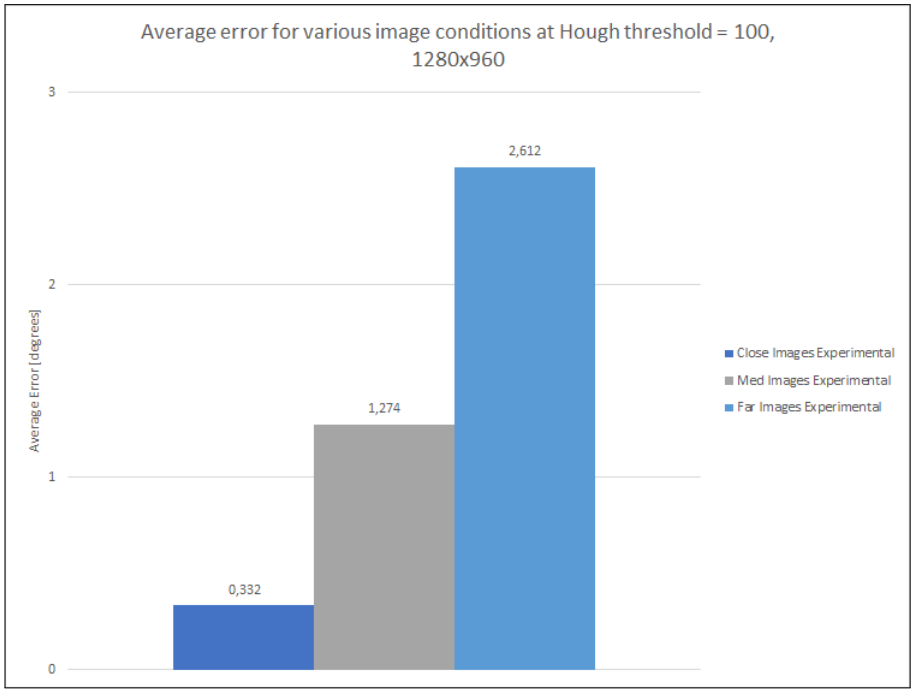


(f) Measured angles for experimental model at Hough threshold 150 for 1600x1200

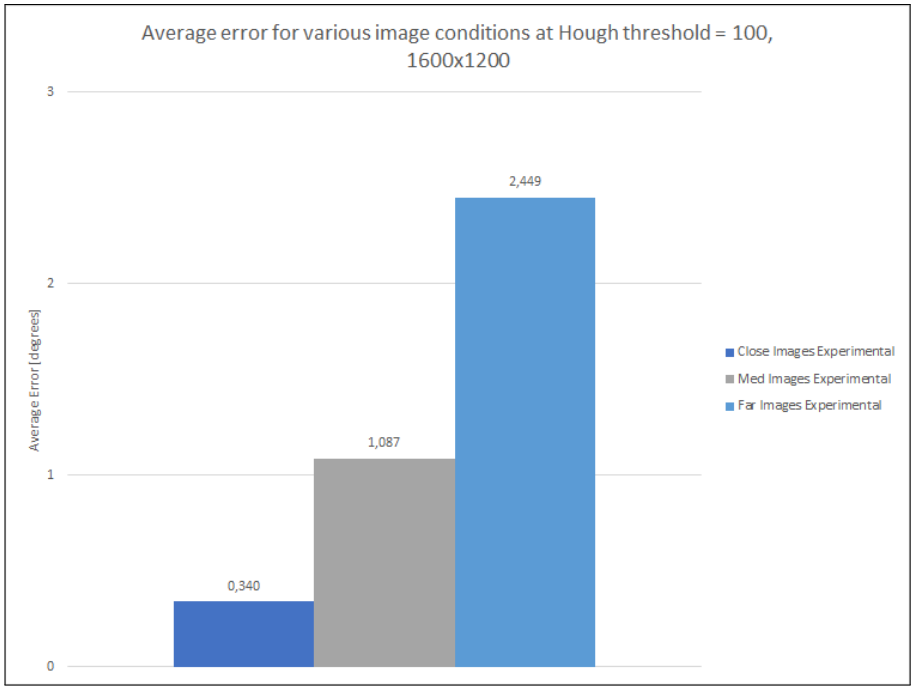
Figure C.6: Graphs of angle vs distance at threshold 100 and 150



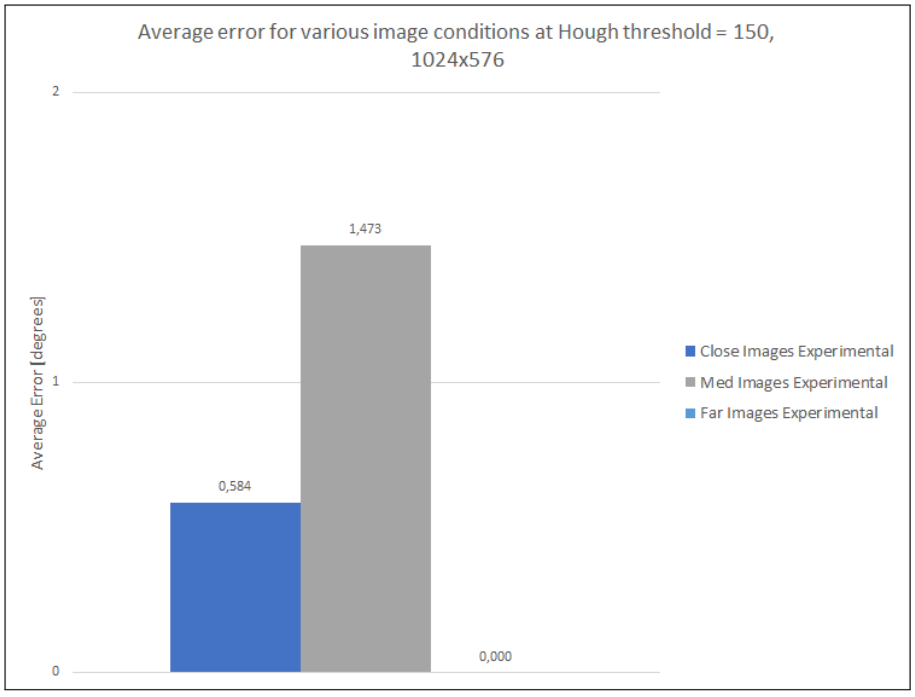
(a) Average error for various image conditions at Hough threshold 100, 1024x576



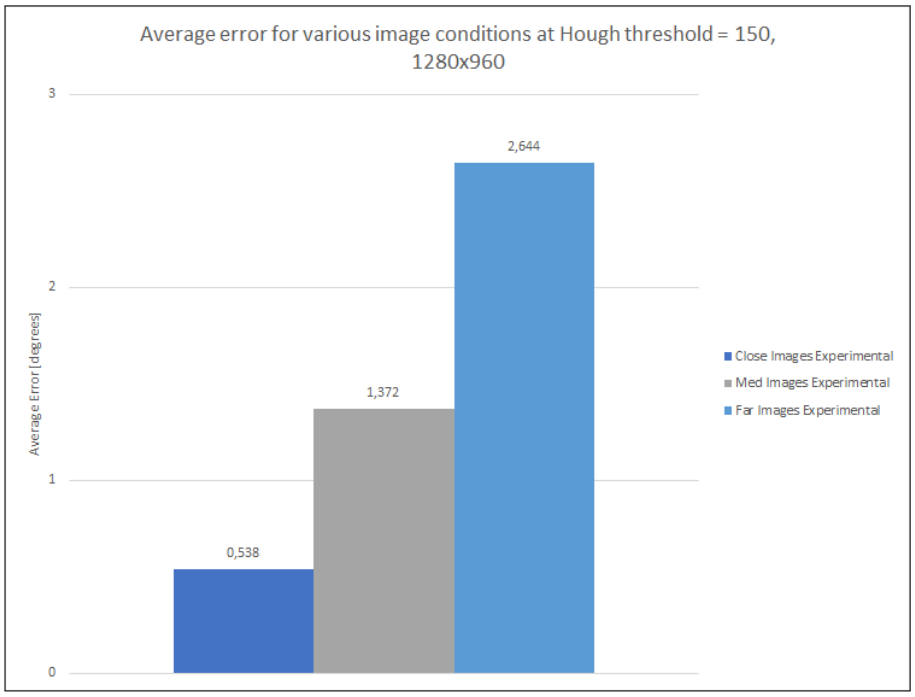
(b) Average error for various image conditions at Hough threshold 100, 1280x960



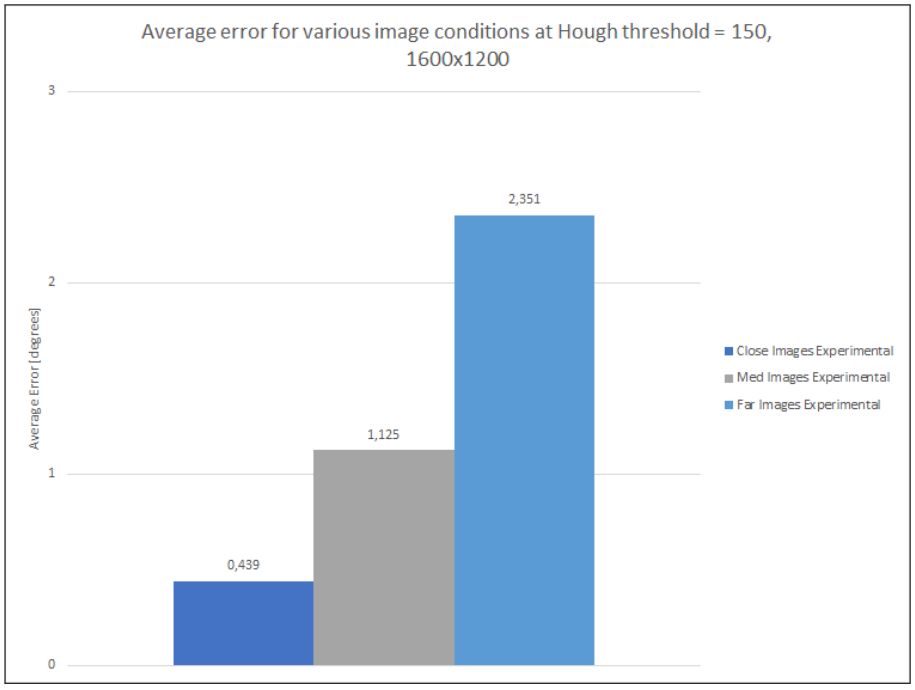
(c) Average error for various image conditions at Hough threshold 100, 1600x1200



(d) Average error for various image conditions at Hough threshold 150, 1024x576

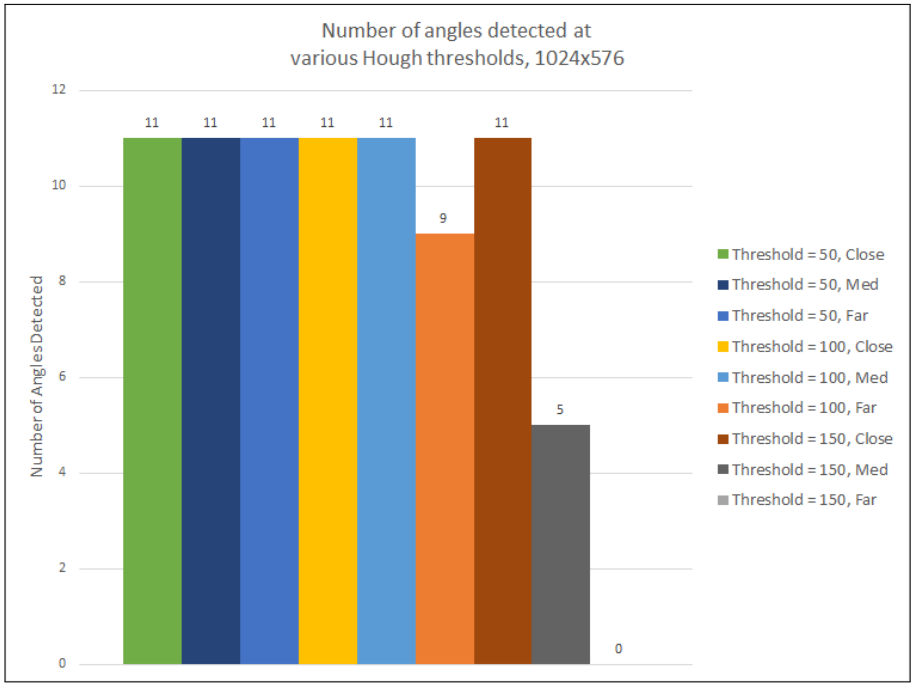


(e) Average error for various image conditions at Hough threshold 150, 1280x960

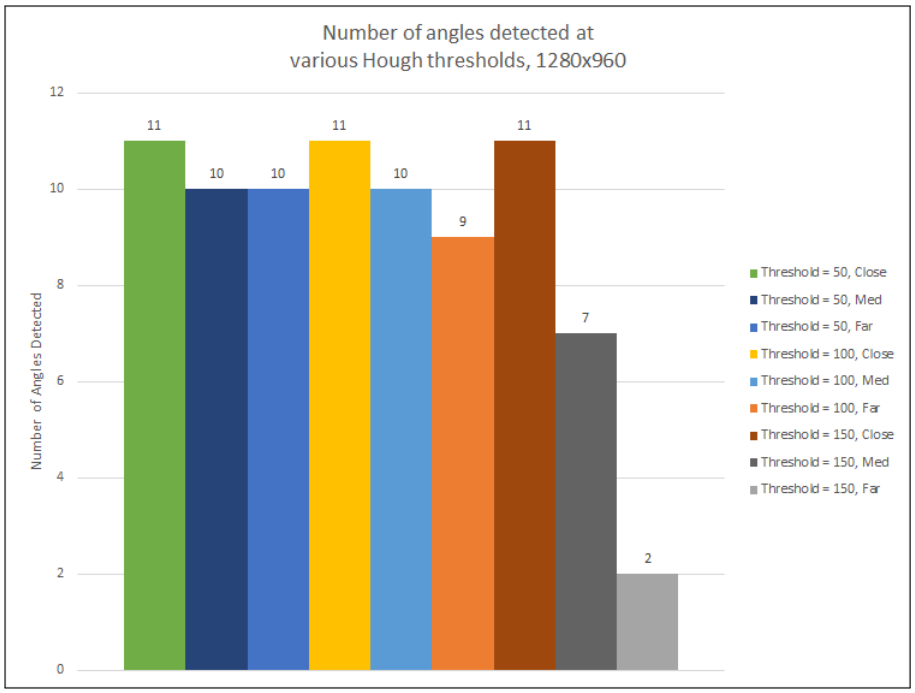


(f) Average error for various image conditions at Hough threshold 150, 1600x1200

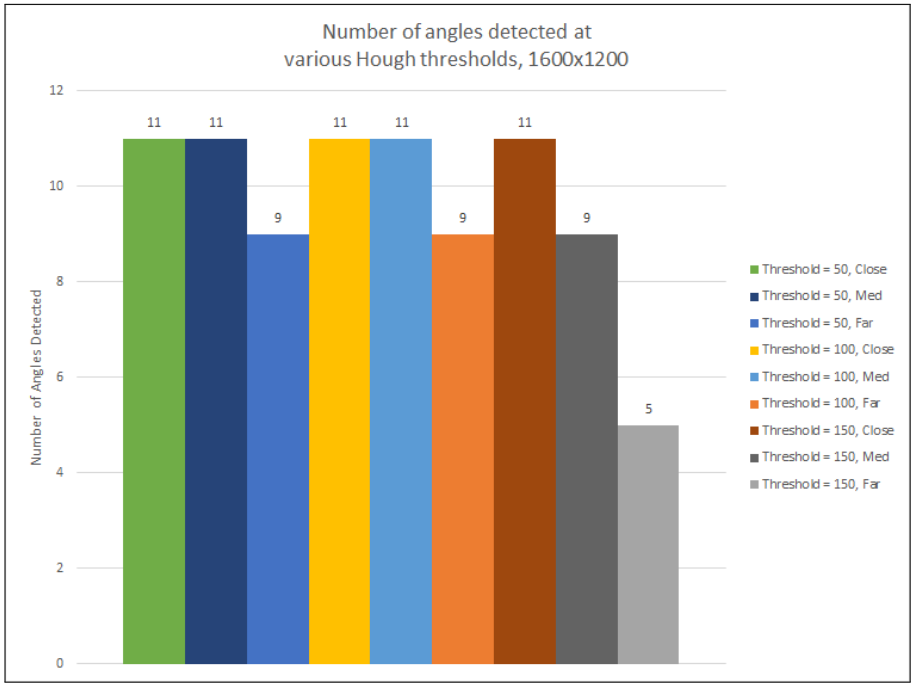
Figure C.7: Graphs of average error vs distance at threshold 100 and 150



(a) Number of angles detected at various Hough thresholds, 1024x576. A higher Hough threshold produces less lines detected, especially at further distances



(b) Number of angles detected at various Hough thresholds, 1280x960. A higher Hough threshold produces less lines detected, consistent with the data for low resolution



(c) Number of angles detected at various Hough thresholds, 1600x1200. A higher Hough threshold produces less lines detected for all the resolutions, but produces the least amount of lines when furthest away from the edge

Figure C.8: Graphs of number of angles detected for various Hough thresholds, analysing the effects of Hough threshold